SOCIAL COMPUTING FOR PERSONALIZATION AND CREDIBLE INFORMATION MINING USING PROBABILISTIC GRAPHICAL MODELS

A Thesis
Presented to
The Academic Faculty

by

Jun Zou

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2016

SOCIAL COMPUTING FOR PERSONALIZATION AND CREDIBLE INFORMATION MINING USING PROBABILISTIC GRAPHICAL MODELS

Approved by:

Professor Faramarz Fekri, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Mark A. Davenport
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Steven W. McLaughlin
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Justin Romberg
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Yajun Mei
School of Industrial and Systems Engineering
Georgia Institute of Technology

Date Approved: July 18, 2016

To my parents,

their love and support made this possible.

ACKNOWLEDGEMENTS

I am deeply grateful to my PhD advisor Prof. Faramarz Fekri for guiding me to pursue the exciting and challenging research in this dissertation. His kind support and encouragement led me to the achievements. I am also very thankful to other dissertation committee members: Professors Mark A. Davenport, Steven W. McLaughlin, Justin Romberg, and Yajun Mei, for agreeing to serve on the committee. I thank the diligent staff members at Center for Signal and Information Processing: Patricia Dixon, Raquel M. Plaskett, and Christopher B. Thomas.

I would like to thank former and current fellow labmates in Prof. Fekri's research group. Thank Dr. Erman Ayday and Arash Einolghozati for introducing me to the recommender system project. Thank Afsin Abdi, Dr. Ahmad Beirami, Ali Payani, Dr. Mohsen Sardari, Yashas Malur Saidutta, Dr. Xin Tian, and Hang Zhang. I was so fortunate to work together with them in an intellectually stimulating environment.

I owed a lot to many dearest friends and my parents. They always loved me and cared for me throughout my PhD study.

The research work in this dissertation was supported by National Science Foundation under Grant No. IIS-1115199, and a gift from the Cisco University Research Program Fund.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In this dissertation, we address challenging social computing problems in personalized recommender systems and social media information mining. We tap into probabilistic graphical models, including directed and undirected graphical models, to model a large number of observed and unobserved variables as well as various dependency relationships between variables, and develop efficient computation algorithms that exploit the graph structure to solve the problems.

In recommender systems, we propose probabilistic graphical models for Collaborative Filtering (CF) algorithms in various problem settings, and solve them using Belief Propagation (BP) algorithms that allow scalable and distributed implementations. Firstly, user similarities are computed in factor graphs. Then unknown ratings are predicted in Pairwise Markov Random Fields (PMRFs). Further, when online social networks of users are provided, a Bayesian Network (BN) recommendation system is constructed based on user relations to improve recommendation for cold-start users or users do not have sufficient ratings. To preserve user privacy, a semi-distributed item-based CF system is developed, which employs semi-distributed BP for item similarity computation in factor graphs, without disclosing ratings to the server or other peer users. Finally, to protect CF recommender systems from shilling attacks, a factor graph is proposed to jointly detect colluding spammers, which significantly improves detection accuracy over classification algorithms based on a single user's rating patterns.

In social media information mining, to detect false information and keep track of information credibility, we propose a generative probabilistic model to predict the

credibility of events in Twitter-like social media using streaming tweets. The proposed algorithm predicts credibility much faster than existing offline algorithms and updates prediction online with newly observed tweets. Further, to identify suspicious users that perform malevolent activities such as spamming and phishing, we propose a probabilistic PMRF model for predicting the trustworthiness of social media users. The PMRF model improves prediction accuracy by taking into account user relationships compared to existing prediction algorithms for individual users.

CHAPTER I

INTRODUCTION

This dissertation centers on the novel application of probabilistic graphical models on two areas: (1) Personalized recommender systems; (2) Credible information mining.

1.1 Personalized Recommender Systems

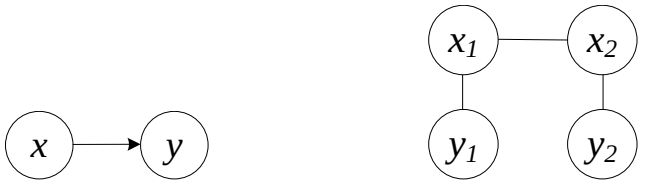
The thriving of the Internet and online services has overwhelmed users with an explosive amount of information, e.g., hundreds of thousands of movies on Netflix.com [4, 65]. The downside of the growth of this is that it is nearly impossible for users to find the items, e.g., books and movies, that interest them. Recommender systems are powerful tools for providing personalized recommendations of items that meet user preferences, which improve user satisfaction and reduce user effort [95, 96]. Recommender system algorithms can be generally classified into two categories: the content-based approach and the Collaborative Filtering (CF) approach [3]. The content-based approach requires explicit interest profiles for users and items, and makes recommendations by matching user profiles with item profiles [11, 78, 86]. However, this approach is difficult to apply in practice, because (i) users either do not like to disclose personal information, or are not completely aware about their interests, and (ii) it is limited by content analysis techniques, e.g., it is difficult to explicitly describe multimedia content using features. Alternatively, the CF approach exploits the past transactions and user ratings to predict user preferences on unseen items [20, 44, 17, 100]. Many online service providers have adopted CF algorithms, e.g., Amazon.com and Netflix.com [65, 4]. In this dissertation, we mostly focus on CF recommender systems.

As recommender systems are often used in large-scale e-commerce websites with

hundreds of thousands of books and movies, there is a need to develop recommendation algorithms with high computational efficiency for large-scale deployments. Additionally, while personalized recommender systems at e-commerce websites have been very successful, they are faced with important security and privacy challenges: Recommender systems are vulnerable to shilling attacks [58, 28, 77], where a group of spammers are injected into the system and they collaborate to manipulate the recommendations; and users are increasingly concerned with online privacy [1, 92, 14, 24, 76], as they have to disclose their personal data to receive satisfactory recommendation services, and they have no control over how their personal information will be disseminated and used. Our work based on graphical models is also motivated by the need to develop recommender systems suitable for privacy-preserving implementation.

1.2 Credible Information Mining

Online social media services like Twitter greatly facilitate the dissemination of information [56, 16, 27, 30]. In Twitter, users can conveniently self-report activities and stories happening around them by posting messages/statuses known as "tweets" using personal computing devices like smart phones, and the tweets can be read and retweeted by other users. The large number of social media users often results in a large amount of social messages reporting real-world events [56]. Monitoring social media streams, e.g., tweets in Twitter, becomes an effective way to detect events and monitor emergent situations [98, 112, 71]. However, social media is also increasingly exploited to spread rumors and false information [26, 74, 42, 93], e.g., fake images during Hurricane Sandy [41]. False rumors in social media can potentially reach millions of people in short amount of time. Counter measures are thus needed to curb false information from undermining the functionality and utility of social media.



(a) A directed graphical model.

(b) An undirected graphical model.

Figure 1: Illustration of probabilistic graphical models.

Moreover, due to the open nature of social media platforms, there are an increasing number of malicious users or spammers [60, 118, 106, 126], who misuse social networks to perform malevolent activities [21, 48] such as spamming, phishing, and spreading computer viruses. Therefore, there is an urgent need to develop algorithms to effectively predict the trustworthiness of users in social networks, so as to identify suspicious users and limit their activities or suspend their accounts.

1.3 Probabilistic Graphical Models

In all research thrusts in this dissertation, there are a large number of variables (e.g., user similarities in recommender systems), and there are also various relationships between variables (e.g., users who rated a given item). To model the uncertainty in unobserved variables and the relationships between different variables, we tap into probabilistic graphical models [12, 52, 110, 38], which are graphs that represent random variables as nodes and relationships between variables as edges between nodes, compactly expressing joint probability distributions of many variables. Graphical models allow us to decompose a highly complex system into many simpler parts with edges connecting related parts, and their graphical structures provide a factorization which often enables the design of computationally efficient algorithms [87, 54, 80, 122]. These properties make graphical models a very suitable framework for addressing our research problems and developing efficient scalable algorithms.

There are two kinds of probabilistic graphical models: directed graphical models

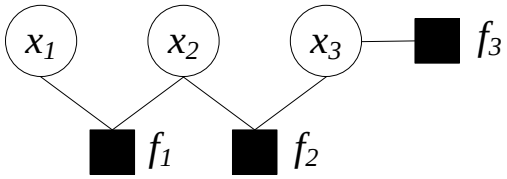


Figure 2: A factor graph for the factorization of $g(x_1, x_2, x_3)$.

and undirected graphical models. In directed graphical models, also known as generative probabilistic models and Bayesian networks [12, 87, 62], a directed edge from variable node x to y indicates that x causes y. As an example illustrated in Fig. 1a, a fire breakout in a building (x = True) causes the fire alarm to sound (y = True), and the joint probability of x and y can be expressed as $P(x, y) = P(y|x)P(x)$. In undirected graphical models, also known as Markov Random Fields (MRFs) [36, 37, 122], an undirected edge connects two related variable nodes x_i and x_j. A simple pairwise MRF model is illustrated in Fig. 1b, which expresses the joint distribution as a factorization

$$P(x_1, x_2, y_1, y_2) = \frac{1}{Z}\psi(x_1, x_2)\phi_1(x_1, y_1)\phi_2(x_2, y_2),$$

where $\psi(x_i, x_j)$ and $\phi_j(x_i, y_i)$ are potential functions, and Z is a normalization factor. Besides the two categories of probabilistic graphical models, a Factor Graph can be used to represent the factorization of a joint distribution function as a bipartite graph [54], where variable nodes and factor nodes represent variables and local functions, respectively, and an edge connects a variable node to a factor node if and only if the variable is an argument of the local function. For example, given the following factorization of a function $g(x_1, x_2, x_3)$,

$$g(x_1, x_2, x_3) = f_1(x_1, x_2)f_2(x_2, x_3)f_3(x_3),$$

the corresponding factor graph is shown in Fig. 2. Both directed and undirected graphical models can also be represented as factor graphs.

1.4 Dissertation Outline

In Chapter 2, we develop probabilistic graphical models and inference algorithms for Collaborative Filtering (CF) recommender systems. Firstly, we propose to compute user similarity on appropriately chosen factor graphs for the neighborhood CF method. The key component for neighborhood methods is the computation of similarities between users. There are several well-known similarity computation methods including Vector Space Similarity (VSS) [19] and Pearson Correlation Coefficient (PCC) [99]. However, the accuracy of those methods is not satisfactory. The proposed factor graph approach improves the accuracy of the neighborhood method by properly modelling the joint distribution of user similarities. Secondly, we propose to predict unknown item ratings on Pairwise Markov Random Fields (PMRFs). The neighborhood method is based on either user similarity or item similarity, whereas the PMRF model allows to exploit both user similarity and item similarity for prediction. For both factor graphs and PMRFs, we apply the Belief Propagation (BP) algorithm which operates on the probabilistic graphical models to exploit the factorization to perform inference efficiently. We evaluate the rating prediction accuracy of the proposed algorithms using 100K MovieLens dataset.

In Chapter 3, we propose a Bayesian Network (BN) based social recommendation algorithm to generate recommendations using implicit user preference propagation over social networks. CF based recommender systems require users to provide historic rating data, which is not suitable for new users or users who do not provide sufficient number of ratings. In social networks, people are more likely to connect to other people sharing similar interests, and they are influenced more by their social connections [113]. By exploiting the social structure of users, social recommender systems can make satisfactory recommendations even for cold-start users. We construct the BN model based on the user relations on social networks, and develop an Expectation Propagation (EP) message-passing algorithm. We evaluate the top-N

recommendation performance on the Epinions dataset.

In Chapter 4, we propose a semi-distributed item-based CF approach to privacy-preserving recommender systems. In traditional recommender systems, the recommendation algorithm is run at the central server of the commercial websites or other third party providers, and thus users have to disclose their personal information and ratings to the server. Users' rating data can be used to infer user demographics, such as age and gender [114], and even uncover user identities and reveal sensitive personal information with access to other databases [81]. Unfortunately, users have no control over how their personal data will be disseminated and used [1]. Most existing privacy-preserving recommender systems are developed either by obfuscating user data with random noise [91] or fake data [104] at the cost of recommendation accuracy, or by encrypting user data using cryptographic techniques [24] which require careful key management. We propose a recommender system with an intrinsic privacy-preserving property. We construct factor graphs for item similarity computation, and develop a semi-distributed BP algorithm without directly exposing user ratings to the server or other peer users. We analyze the privacy of the semi-distributed BP from an information-theoretic perspective, and also evaluate the recommendation performance on the MovieLens dataset.

In Chapter 5, we propose a probabilistic factor graph model for detecting spammers that launch shilling attacks on CF recommender systems. In shilling attacks, a group of spammers collaborate to manipulate the recommendations for their benefit [58], e.g., to recommend their products more often, by injecting spam ratings. Existing detection algorithms [28, 22] extract features from user ratings to detect the spammers. Those algorithms suffer from low accuracy, as they only look at individual user rating patterns. We propose a factor graph model that exploits the collaborative spamming behaviors among spammers to jointly detect them. We evaluate the detection performance using simulated shilling attacks on MovieLens dataset, and show

that as the number of target items increases, the detection precision and recall of the proposed algorithm improves significantly.

In Chapter 6, we propose social link recommendation algorithms using both popularity and similarity. Due to the huge number of users in social media, recommendation algorithms are needed to help users automatically discover new social links to follow/connect. Existing link recommendation algorithms focus on link structure [64] or "weighted popularity". They are suitable for purposes like establishing social connections. However, social media like Twitter are also adopted by users to receive information that interests them. Hence, it is promising to exploit the similarity between Twitter users for recommendation [75]. We propose two approaches to exploiting both popularity and similarity: rank aggregation and popularity-biased Bayesian personalized ranking. We evaluate the link prediction performance on real-world online social network datasets, and show that the proposed algorithms can combine the advantages of popularity-based and similarity-based algorithms to improve performance.

In Chapter 7, we propose a generative probabilistic model for real-time event credibility prediction in Twitter-like social media. While online social media services greatly facilitate the dissemination of information, they are also increasingly exploited to spread rumors and false information. Existing methods for assessing event credibility in social media require various aggregation features extracted from a set of tweets related to an event that are collected over a long period of time [26, 57]. This can cause significant delays in the credibility prediction task from the time the event first takes place. We develop a fast online prediction algorithm using streaming tweets based on a generative probabilistic model. We evaluate both the offline batch prediction and online streaming prediction performance of the proposed model using real social media events collected from Twitter. The results show that our model outperforms other algorithms based on aggregation analysis, and the online streaming

prediction performance quickly approaches that of the batch prediction with only a few hundred tweets.

In Chapter 8, we propose a probabilistic Pairwise Markov Random Field (PMRF) model for predicting the trustworthiness of social media users. Spammers misuse social networks to perform malevolent activities such as spamming and phishing. Thus, effective algorithms are needed to identify suspicious users and limit their activities or suspend their accounts. Existing works [106, 117, 60] introduced feature-based classification methods that capture behavioural characteristics of individual users, but they ignore the relationships between users. The establishment of bidirectional connections between users often indicates some degree of similarity in their trustworthiness. The proposed PMRF model takes into account both user features and social relationships. We evaluate the accuracy of the proposal algorithm for predicting spammers using Twitter datasets. The results show that it outperforms other classification algorithms using only features of individual users.

In Chapter 9, we conclude this dissertation and discuss future research.

CHAPTER II

PROBABILISTIC GRAPHICAL MODELS FOR COLLABORATIVE FILTERING

To tackle those challenges in recommender systems as discussed in Section 1.1 of Chapter 1, we develop probabilistic graphical models and efficient inference algorithms for Collaborative Filtering (CF) recommender systems. Firstly, we propose to compute user similarity on appropriately chosen factor graphs [127]. Secondly, we propose to represent a recommender system on a Pairwise Markov Random Field (PMRF) to exploit both user similarity and item similarity for predicting unknown ratings [6]. In both of the probabilistic inference problems, we compute the marginal distributions of similarity/rating variables from their joint distributions given the observed ratings. However, direct computation of the marginal distribution functions is computationally prohibitive for large scale recommender systems. Factor graphs and PMRFs express the factorization of the joint distribution functions. Therefore, we apply the Belief Propagation (BP) algorithm which operates on the probabilistic graphical models to exploit the factorization to perform inference efficiently.

We assume a set of M users, denoted by $\mathbb{U} = \{1, \ldots, M\}$, and a set of N items, denoted by $\mathbb{I} = \{1, \ldots, N\}$, in the recommender system. Specifically, user u provides feedback on item i in the form of rating r_{ui}. Let U_i denote the set of users who have rated item i, and I_u denote the set of items rated by user u. Let $\mathbb{R}$ denote the set of all collected observed ratings from users on items. The CF recommender system takes as input the historic ratings in $\mathbb{R}$, and generates rating predictions for an active user u on unseen items in $\mathbb{I} \backslash I_u$.

2.1 Similarity Inference on Factor Graphs

One of the major approaches for designing recommender systems is Collaborative Filtering (CF) [2]. The CF approach takes as input the historic ratings on items given by users, and predicts ratings on unseen items for each active user. It includes model-based methods and memory-based methods. The model-based method generates rating predictions from a model learned from the collected historic ratings, but the learning process is often time-consuming, which is not suitable for systems with frequent updates. The memory-based method, *a.k.a.* the neighborhood method, can be further divided into user-based and item-based methods. The user-based method recommends to an active user new items favorably rated by other users with similar tastes to the active user [95]. The item-based method on the other hand analyzes the similarity between items using the aggregated user ratings, and recommends to an active user new items that are similar to the items he liked in the past [99].

The key component for neighborhood methods is the computation of similarities between users or items. In this work, we formulate the similarity computation as a probabilistic inference problem of computing the marginal distributions of similarity variables from their joint posterior distribution given the observed ratings [127]. To efficiently solve this problem, we introduce a factorization of the joint distribution in an appropriate chosen factor graph, and apply the Belief Propagation (BP) algorithm [54] that operates in the factor graph to exploit the factorization for efficient inference. In addition, a complexity reduction technique is proposed to contain the exponential increase in computational complexity caused by the high degree at the factor node in our setup of recommender systems. This BP-based approach was motivated by the successful application of BP for iterative decoding algorithms in error-control systems [53]. BP is very efficient in computing marginal functions from global functions of many variables. Moreover, BP can even be applied in situations where exact solutions for marginal functions are computationally intractable, such as the iterative decoding

of Low-Density Parity-Check (LDPC) codes and turbo codes. As we shall see, the concerned problem in this work also belongs to such intractable cases.

The recent applications of BP to CF systems were also introduced in [105, 51, 8, 7]. [105] and [51] proposed message-passing algorithms for performing probabilistic low-rank matrix factorization to represent users and items with low-dimension vectors. [8] and [7] proposed to directly predict ratings in the factor graph, where probabilistic messages on unknown ratings are iteratively exchanged, whereas in this work we focus on inferring the similarities, based on which the unknown ratings are predicted by the neighborhood method.

2.1.1 Background on Neighborhood-based Collaborative Filtering

The neighborhood method can be either user-based [95] or item-based [99]. To predict the rating r_{ui} for user u on an unseen item i, the user-based algorithm sorts the users in U_i according to their similarity to user u in descending order, and finds a subset of top K most similar users, denoted by $\mathcal{N}_{ui}$, where $|\mathcal{N}_{ui}| = K$. We refer to $\mathcal{N}_{ui}$ as the neighbourhood of user u for predicting rating on item i, and K as the neighborhood size. Then r_{ui} is predicted by

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in \mathcal{N}_{ui}} s_{uv} \times (r_{vi} - \bar{r}_v)}{\sum_{v \in \mathcal{N}_{ui}} |s_{uv}|}, \tag{1}$$

where s_{uv} is the similarity between users u and v, and $\bar{r}_u$ is the average rating by user u on all items it has rated in the past. Alternatively, the item-based algorithm predicts r_{ui} as

$$\hat{r}_{ui} = \frac{\sum_{j \in \mathcal{I}_{ui}} s_{ij} \times r_{uj}}{\sum_{j \in \mathcal{I}_{ui}} |s_{ij}|}, \tag{2}$$

where s_{ij} denotes the similarity between items i and j, and $\mathcal{I}_{ui}$ represents a subset of K items in I_u that are deemed most similar to item i.

The similarity computation plays a pivotal role in determining the accuracy of the neighborhood methods. There are several well-known similarity computation methods including Vector Space Similarity (VSS) [19] and Pearson Correlation Coefficient

(PCC) [99]. It is observed that PCC performs better than VSS, as PCC takes into account the difference in average ratings. In the case of user similarity, PCC computes s_{uv} between users u and v as

$$s_{uv} = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}, \tag{3}$$

where $I_{uv} = I_u \cap I_v$ denotes the subset of common items rated by both users u and v. The item similarity can be similarly computed using PCC. The PCC algorithm gains popularity as it is easy to implement and its computational complexity is low, while providing reasonable recommendation performance. In the following, we will develop a BP-based algorithm for similarity computation to improve the accuracy of the neighborhood method, yet with a computational complexity comparable to that of PCC.

2.1.2 Probabilistic Modeling of User Similarity

We focus on the similarity computation for the user-based neighborhood method, which generates rating predictions via (1). Its extension to the item-based neighborhood method can be likewise developed. To predict ratings for active user u using (1), we need to first compute the user similarities between user u and other users. Let $\mathbb{S}_u$ denote the set of concerned user similarities for user u, $\mathbb{S}_u = \{s_{uv} : 1 \leq v \leq M,\, v \neq u\}$.

We model s_{uv} as a discrete random variable that takes values from a predefined alphabet set $\mathcal{S}$ with size $L = |\mathcal{S}|$. We denote $P(\mathbb{S}_u|\mathbb{R})$ as the joint posterior probability distribution of all variables in $\mathbb{S}_u$ given the evidence of observed ratings in $\mathbb{R}$. Then to compute the individual user similarity s_{uv}, we need to find its marginal posterior distribution $P(s_{uv}|\mathbb{R})$, which can be derived as

$$P\left(s_{uv}|\mathbb{R}\right) = \sum_{s_{u1} \in \mathcal{S}} \cdots \sum_{s_{u(v-1)} \in \mathcal{S}} \sum_{s_{u(v+1)} \in \mathcal{S}} \cdots \sum_{s_{uM} \in \mathcal{S}} P\left(\mathbb{S}_u|\mathbb{R}\right). \tag{4}$$

For notational convenience, we rewrite (4) for the sum over all variables in $\mathbb{S}_u$ except

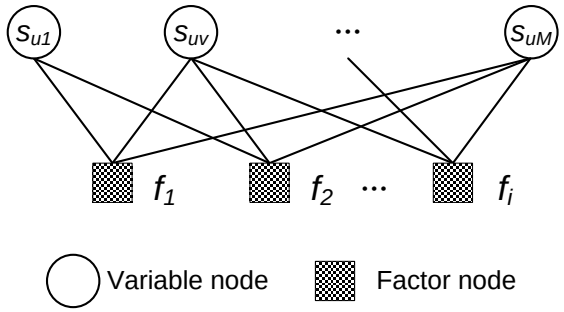


Figure 3: The factor graph $\mathcal{G}_u$ for similarity computation.

s_{uv} as

$$P\left(s_{uv}|\mathbb{R}\right) = \sum_{\mathbb{S}_u \backslash s_{uv}} P\left(\mathbb{S}_u|\mathbb{R}\right). \tag{5}$$

Similar notations are used for this type of sum throughout this paper. Unfortunately, the complexity of direct computation using (5) is $\mathcal{O}\left(L^M\right)$, which is exponential in the number of users. We therefore propose to factorize the joint distribution $P(\mathbb{S}_u|\mathbb{R})$ into local functions on a factor graph, which allows the application of the efficient BP algorithm for inferring marginal distributions.

2.1.2.1 Modeling Similarity via Factor Graphs

A factor graph is a bipartite graph that expresses the factorization structure of a function, where variable nodes and factor nodes represent variables and local functions, respectively, and an edge connects a variable node to a factor node if and only if the variable is an argument of the local function represented by the factor node [54]. To construct a factor graph for $P(\mathbb{S}_u|\mathbb{R})$, we first find a proper factorization of $P(\mathbb{S}_u|\mathbb{R})$. We notice that dependencies among user similarities are induced by user ratings on the common items. Hence, for each item $i \in I_u$, we let $\mathbb{S}_{ui} = \{s_{uv} : v \in U_i \backslash u\}$ be the subset of user similarities between user u and the users who have rated item i, and use a local function $f_i\left(\mathbb{S}_{ui}\right)$ to model the dependencies among variables in $\mathbb{S}_{ui}$ based on observed ratings on item i. Then the factorization of $P(\mathbb{S}_u|\mathbb{R})$ can have the

following form

$$P(\mathbb{S}_u|\mathbb{R}) = \frac{1}{Z}\prod_{i\in I_u} f_i\left(\mathbb{S}_{ui}\right), \tag{6}$$

where Z is a normalization constant.

Now we construct a factor graph $\mathcal{G}_u$ for $P(\mathbb{S}_u|\mathbb{R})$ according to (6) as illustrated in Fig. 3. Each variable s_{uv} is represented by a variable node v, and each local function $f_i\left(\mathbb{S}_{ui}\right)$ is represented by factor node i. The variable nodes in $\mathcal{V}_i = U_i \backslash u$ are connected to factor node i via edges, and thus the degree at each factor node i is $|\mathcal{V}_i|$.

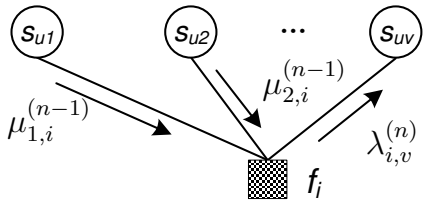
The local function f_i at factor node i is designed specifically for the eventual goal, which is to predict ratings using (1). Assuming the rating r_{ui} on item i in U_i is unknown, we predict r_{ui} as

$$\hat{r}_{ui}\left(\mathbb{S}_{ui}\right) = \bar{r}_u + \frac{\sum_{v\in\mathcal{V}_i} s_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v\in\mathcal{V}_i} |s_{uv}|}. \tag{7}$$

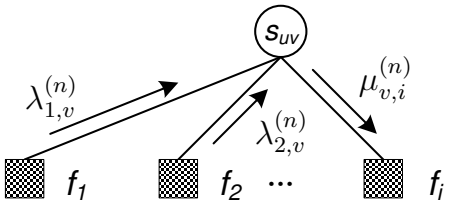
Note that (7) has a similar form to (1), except that the observed ratings on item i from all users in $\mathcal{V}_i$ are used in (7), since user similarities are not determined yet. We then define the factor node function f_i as

$$f_i\left(\mathbb{S}_{ui}\right) = \frac{1}{Z_i}\exp\left\{-\frac{1}{\sigma^2}\left(\hat{r}_{ui}\left(\mathbb{S}_{ui}\right) - r_{ui}\right)^2\right\}, \tag{8}$$

where Z_i is a normalization constant, and σ^2 is a designing parameter which controls the sensitivity of the function to the discrepancy between $\hat{r}_{ui}\left(\mathbb{S}_{ui}\right)$ and the actual rating r_{ui}. The factor node function f_i here can be seen as a soft check constraint, which checks the weighted average rating in (7) against the true rating r_{ui} for a given configuration of user similarities in $\mathbb{S}_{uv}$. f_i decreases as the discrepancy between $\hat{r}_{ui}$ and r_{ui} increases. It is insightful to compare our factor graph formulation to iterative BP decoding of LDPC codes. Indeed, the factor node in our setup plays a similar role as the check node in LDPC decoding.



(a) λ-message $\lambda_{i,v}^{(n)}$ from factor node i to variable node v.



(b) μ-message $\mu_{v,i}^{(n)}$ from variable node v to factor node i.

Figure 4: Iterative message-passing on a factor graph.

2.1.3 Iterative Message Passing for Similarity Computation

Our goal is to compute the marginal posterior probability distribution $P\left(s_{uv}|\mathbb{R}\right)$ from $P(\mathbb{S}_u|\mathbb{R})$ for each s_{uv} in $\mathbb{S}_u$. We tap into the BP algorithm for efficient inference on the factor graph. Due to loops in the constructed factor graph $\mathcal{G}_u$, we need to apply the "loopy" BP algorithm, which performs iterative message exchanging between variable nodes and factor nodes. Although its computed results are not exact, "loopy" BP has shown great success in applications such as LDPC decoding, in which the underlying factor graph also has loops, as it achieves near optimal performance with linear complexity.

We define two types of messages following the principle of the sum-product BP algorithm [54]: (i) The λ-message $\lambda_{i,v}(s_{uv})$ sent from a factor node i to a variable node v, and (ii) the μ-message $\mu_{v,i}(s_{uv})$ sent from a variable node v to a factor node i. The λ-messages and μ-messages are iteratively updated and passed along edges in the factor graph. In the n-th iteration, to compute $\lambda_{i,v}^{(n)}(s_{uv})$, factor node i multiplies local function f_i with all μ-messages received in the last iteration except the one from

the recipient variable node v, and sums out variables in $\mathbb{S}_{ui}$ excluding s_{uv} as below

$$\lambda_{i,v}^{(n)}(s_{uv}) \propto \sum_{\mathbb{S}_{ui} \setminus s_{uv}} f_i(\mathbb{S}_{ui}) \prod_{w \in \mathcal{V}_i \setminus v} \mu_{w,i}^{(n-1)}(s_{uw}). \tag{9}$$

Figure 4a illustrates the computation of λ-message $\lambda_{i,v}(s_{uv})$. The λ-message $\lambda_{i,v}^{(n)}(s_{uv})$ tells user v the likelihoods of $s_{uv} = s, \forall s \in \mathcal{S}$. It says given how similar other users' ratings on item i are to user u's, how similar user v is to user u from item i's point of view.

Then the algorithm continues to update μ-messages using the λ-messages generated in the current iteration. To obtain $\mu_{v,i}^{(n)}(s_{uv})$, variable node v computes the product of all incoming λ-messages except the one from the recipient factor node i as follows

$$\mu_{v,i}^{(n)}(s_{uv}) \propto \prod_{j \in F_v \setminus i} \lambda_{j,v}^{(n)}(s_{uv}) \tag{10}$$

where F_v denotes the set of factor nodes connected to variable node v. Here, $F_v = I_v \cap I_u$. Figure 4b illustrates the computation of μ-message $\mu_{v,i}(s_{uv})$. The μ-message $\mu_{v,i}^{(n)}(s_{uv})$ tells item i the probabilities of $s_{uv} = s, \forall s \in \mathcal{S}$. It says given how similar user v is to user u from other items' point of view, how similar user v's rating is to user u's on item i.

In each iteration, the messages are computed for each node in the factor graph. To check convergence of the algorithm, we compute the marginal probability distributions after each iteration as follows

$$P(s_{uv}|\mathbb{R}) = \frac{1}{Z_v} \prod_{i \in F_v} \lambda_{i,v}^{(n)}(s_{uv}), \tag{11}$$

where Z_v is a normalization constant. The algorithm exits iteration when $P(s_{uv}|\mathbb{R})$'s converge. Finally, we can estimate user similarity s_{uv} from (11) according to various criteria. We consider the minimum mean squared error criterion here, and thus the optimal estimation of user similarity s_{uv} is given by its expectation

$$\bar{s}_{uv} = \sum_{s \in \mathcal{S}} s P(s_{uv} = s|\mathbb{R}). \tag{12}$$

Algorithm 1 Iterative message passing for similarity computation

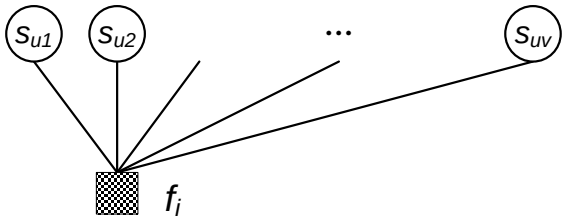
- Initialization. $\mu_{v,i}^{(0)}(s_{uv}) = \frac{1}{|\mathcal{S}|}$ and $n = 0$.
- Iterative message-passing:
  (a) Update λ-message using (9);
  (b) Update μ-message using (10);
  (c) $n = n + 1$. Repeat (a) and (b) until convergence.
- Compute marginal distributions of s_{uv}'s using (11).
- Compute $s_{uv} \in \mathbb{S}_u$ using (12).

We summarize the iterative BP algorithm in Alg. 1.

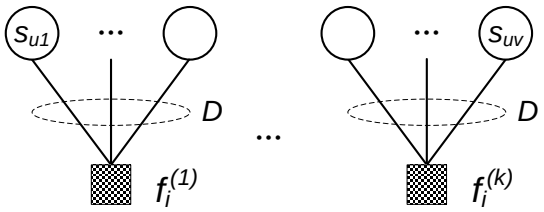
2.1.4 Complexity Reduction

We analyze the complexity of the proposed BP-based algorithm in terms of number of multiplications. The complexity to update a μ-message using (10) is only $\mathcal{O}(|I_u|)$, but the complexity to update a λ-message using (9) is $\mathcal{O}\left(|\mathcal{V}_i|L^{|\mathcal{V}_i|}\right)$, which is exponential in the degree of the factor node. From the construction process of the factor graph $\mathcal{G}_u$ in Sec. 2.1.2.1, we know that $\mathcal{V}_i = U_i \backslash u$. Unfortunately, in recommender systems, an item can be rated by over hundreds of users. Therefore, direct application of the BP algorithm is not feasible. We thus propose to construct a new factor graph $\hat{\mathcal{G}}_u$ from $\mathcal{G}_u$ for complexity reduction as follows.

The key here is to reduce the degree of factor nodes. For a high-degree factor node i, the variable nodes in $\mathcal{V}_i$ are divided into small groups of size D, resulting in $G_i = \lceil \frac{|\mathcal{V}_i|}{D} \rceil$ groups. We denote the subset of variable nodes in group k of factor node i as $\mathcal{V}_i^{(k)}$, $1 \leq k \leq G_i$. For each variable node $v \in \mathcal{V}_i$, we set an indicator $v_i^{(k)} = 1$ if $v \in \mathcal{V}_i^{(k)}$ and $v_i^{(k)} = 0$ otherwise. Note that $\sum_{k=1}^{G_i} v_i^{(k)} = 1$. Further, we connect a separated factor node $i^{(k)}$ to variable nodes in $\mathcal{V}_i^{(k)}$. We do the same for all other factor nodes in the original factor graph $\mathcal{G}_u$. The complexity-reduction technique on



(a) A high-degree factor node in $\mathcal{G}_u$



(b) Multiple low-degree factor nodes in $\hat{\mathcal{G}}_u$

Figure 5: Illustration of complexity-reduction in the factor graph.

a factor graph is illustrated in Fig. 5. Let $\mathbb{S}_{ui}^{(k)} = \{s_{uv} : v \in \mathcal{V}_i^{(k)}\}$. The factor function $f_i^{(k)}$ of factor node $i^{(k)}$ is derived from (8) as

$$f_i^{(k)}\left(\mathbb{S}_{ui}^{(k)}\right) = \frac{1}{Z_i^{(k)}} \exp\left\{-\frac{1}{\sigma^2}\left(\hat{r}_{ui}^{(k)}(\mathbb{S}_{ui}^{(k)}) - r_{ui}\right)^2\right\}, \tag{13}$$

where $Z_i^{(k)}$ is a normalization constant, and

$$\hat{r}_{ui}^{(k)}(\mathbb{S}_{ui}^{(k)}) = \bar{r}_u + \frac{\sum_{v \in \mathcal{V}_i^{(k)}} s_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v \in \mathcal{V}_i^{(k)}} |s_{uv}|}. \tag{14}$$

To better understand the proposed technique for complexity reduction, we provide an intuitive explanation using user-item relations. We divide the users of item i into multiple groups, and create a separated virtual item $i^{(k)}$ for each group k. Users in $\mathcal{V}_i^{(k)}$ give the same ratings on item $i^{(k)}$ as the original item i, i.e., item $i^{(k)}$ can be seen as a copy of item i, but only rated by group k. As in Sec. 2.1.2.1, now the factorization of $P(\mathbb{S}_u|\mathbb{R})$ can be expressed by

$$P(\mathbb{S}_u|\mathbb{R}) = \frac{1}{Z} \prod_{i \in I_u} \prod_{k=1}^{G_i} f_i^{(k)}\left(\mathbb{S}_{ui}^{(k)}\right). \tag{15}$$

And from (15), we can construct a new factor graph which is exactly the complexity reduction factor graph $\hat{\mathcal{G}}_u$.

While the new factor graph $\hat{\mathcal{G}}_u$ is not mathematically equivalent to the original factor graph $\mathcal{G}_u$, they share the same underlying principle, that is, they properly assign probabilities for each configuration of user similarities by checking the weighted rating against the true item rating via factor node functions (or, soft check node constraints). Moreover, we can apply the similar iterative BP algorithm described in Sec. 2.1.3 with minor modifications. The new λ-message $\lambda^{(n)}_{i^{(k)},v}(s_{uv})$ sent from factor node $i^{(k)}$ to variable node v is given by

$$\lambda^{(n)}_{i^{(k)},v}(s_{uv}) \propto \sum_{\mathbb{S}^{(k)}_{ui}\setminus s_{uv}} f^{(k)}_i\left(\mathbb{S}^{(k)}_{ui}\right) \prod_{w\in\mathcal{V}^{(k)}_i\setminus v} \mu^{(n-1)}_{w,i^{(k)}}(s_{uw}). \tag{16}$$

And the new μ-message $\mu^{(n)}_{v,i^{(k)}}(s_{uv})$ sent from variable node v to factor node $i^{(k)}$ is given by

$$\mu^{(n)}_{v,i^{(k)}}(s_{uv}) \propto \prod_{j\in\hat{F}_v\setminus i^{(k)}} \lambda^{(n)}_{j,v}(s_{uv}), \tag{17}$$

where $\hat{F}_v = \left\{j^{(k)} : v^{(k)}_j = 1, j \in I_v \cap I_u, 1 \le k \le G_j\right\}$.

Now the complexity of updating one λ-message is effectively reduced to $\mathcal{O}\left(DL^D\right)$ from $\mathcal{O}\left(|\mathcal{V}_i|L^{|\mathcal{V}_i|}\right)$ by using (16), where the group size D is a small integer and is adjustable, while the total number of λ-messages need to be updated remains the same. The computational complexity with regard to μ-messages does not change. Then to solve for $\mathbb{S}_u$ on $\hat{\mathcal{G}}_u$, the overall complexity for one iteration of the iterative BP algorithm becomes $\mathcal{O}\left(\bar{M}\bar{N}DL^D + M\bar{N}^2\right)$, where $\bar{M}$ is the average number of users of one item, and $\bar{N}$ is the average number of items rated by one user. Since the number of items a user can consume is limited by his time and money, we assume $\bar{N}$ is much smaller than N, and we also assume $\bar{M}$ grows in the order of $M^{1-\epsilon}$, where $0 < \epsilon < 1$. Then we can rewrite the computation complexity on $\hat{\mathcal{G}}_u$ as $\mathcal{O}\left(M^{1-\epsilon}\bar{N}DL^D + M\bar{N}^2\right)$, and when M is large, it is dominated by the second term, so the complexity becomes $\mathcal{O}\left(M\bar{N}^2\right)$.

2.2 Recommender Systems via PMRF

In this work, we solve the recommender system's problem on a Pairwise Markov Random Field (PMRF), where we model proper similarity relationships between items and construct local evidences based on user similarity, using which we compute the marginal probability distribution functions of the variables representing the unknown ratings to be predicted. However, computing the marginal probability functions is computationally prohibitive for large scale recommender systems. Therefore, we propose to utilize the BP algorithm to efficiently compute these marginal probability distributions in linear complexity. The BP-based recommendation algorithm offers a significant advantage on scalability while providing competitive accuracy for the recommender systems.

The neighborhood method aggregates ratings either from other users of the same item, known as user-based methods [95], or from other items rated by the active user, known as item-based methods [99], to predict the unknown rating on an item by a user. Since the user-based method ignores the information from item similarity, and vice versa, the hybrid method is suggested in [111, 67] to combine both user similarity and item similarity. Our proposed approach via PMRF also incorporates both.

Graphical formulations of CF recommender systems using random fields are also studied in [108, 29]. In [108], the authors integrate content-based and CF methods on a conditional Markov random field, where each vertex represents a rating, each edge connects two ratings on the same item or two ratings by the same user, and a local evidence node at each vertex encodes user and item profiles. It requires extensive training to learn the model. In [29], the authors predict item ratings for one active user on a single graph in each run, but they do not make use of user relations.

In [8, 7], the authors proposed recommender systems on a factor graph and introduced a BP-based approach to solve for ratings. This approach was motivated by the successful application of BP for solving reputation systems on factor graphs [9, 10, 5].

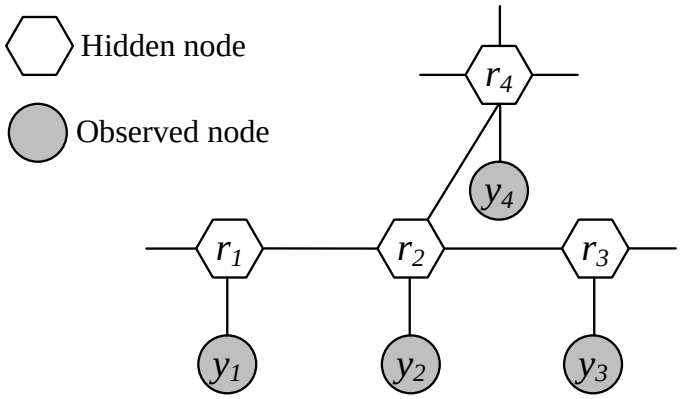


Figure 6: Illustration of the PMRF model for rating prediction.

But the factor graph solution, despite its good performance, only captures the user similarities. By using PMRF, our work models item similarities as edges connecting unknown item ratings and user similarities as local evidences.

2.2.1 Formulation of Recommender Systems on PMRF

Given an arbitrary user z (referred to as the active user), let $\mathbb{R}_z = \{r_{zi} : i \in \mathbb{I} \backslash I_z\}$ be the collection of variables representing the ratings on items to be predicted, where I_z denotes the subset of items whose ratings are already observed. We assume that the rating values are integers from the set $\Upsilon = \{1, 2, 3, 4, 5\}$. We model the unknown ratings $\mathbb{R}_z$ on a PMRF as illustrated in Fig. 6. For each item rating r_{zi} to be predicted, we assign a hidden node, or item variable node, shown as a hexagon, which takes values from a set of discrete ratings in Υ. To incorporate ratings from other users, we connect each item node i to a local evidence node y_{zi} (observed node) shown as a circle, which represents the average item rating from other users weighted by the user similarities between other users and the active user as

$$y_{zi} = \bar{r}_z + \frac{\sum_{v \in \hat{U}_i} s_{zv}(r_{vi} - \bar{r}_v)}{\sum_{v \in \hat{U}_i} |s_{zv}|}, \tag{18}$$

where $\hat{U}_i$ denotes a set of K_z users that are most similar to the active user z among users who have rated item i. Further, the user similarity s_{zv} is computed on a factor graph in Sec. 2.1. To model the compatibility between ratings on similar items, we

use an edge to connect two items i and j if the similarity s_{ij} between them is above a predefined threshold σ_{th} and one of them is among the top K most similar items of the other. Here, we compute item similarities using the adjusted cosine similarity measure [99] as follows

$$s_{ij} = \frac{\sum_{v \in U_{ij}} (r_{vi} - \bar{r}_v)(r_{vj} - \bar{r}_v)}{\sqrt{\sum_{v \in U_{ij}} (r_{vi} - \bar{r}_v)^2} \sqrt{\sum_{v \in U_{ij}} (r_{vj} - \bar{r}_v)^2}}, \tag{19}$$

where $U_{ij} = U_i \cap U_j$.

We define a local evidence function $\phi_i(r_{zi})$ to capture the dependency between r_{zi} and its local evidence y_{zi}. $\phi_i(r_{zi})$ is a probability distribution over possible ratings in Υ. There are many potential choices of designing $\phi_i(r_{zi})$. In this work, we distribute the probability on the integer ratings in Υ close to y_{zi}:

- If $1 \leq y_{zi} \leq 5$, $\lfloor y_{zi} \rfloor \phi_i(r_{zi} = \lfloor y_{zi} \rfloor) + \lceil y_{zi} \rceil \phi_i(r_{zi} = \lceil y_{zi} \rceil) = y_{zi}$;
- If $y_{zi} < 1$, $\phi_i(r_{zi} = 1) = 1$;
- If $y_{zi} > 5$, $\phi_i(r_{zi} = 5) = 1$.

The local evidence function represents how one item should be rated according to the local evidence in (18). We also define a compatibility function between two connected item variable nodes i and j as

$$\psi_{ij}(r_{zi}, r_{zj}) \propto \exp\left\{-\sigma_{ij}^{-2}(r_{zi} - r_{zj})^2\right\} \tag{20}$$

where σ_{ij} is adjusted according to similarity between items i and j. The compatibility function penalizes the configuration of ratings on the pair of connect items if they deviate from each other. The overall joint distribution of unknown item ratings $\{r_{zi} | i \in \mathbb{I} \backslash I_z\}$ represented by the PMRF is

$$P(\{r_{zi}, i \in \mathbb{I} \backslash I_z\} | \mathbb{R}) = \frac{1}{Z} \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(r_{zi}, r_{zj}) \prod_{i \in \mathbb{I} \backslash I_z} \phi_i(r_{zi}), \tag{21}$$

where $\mathcal{E}$ denotes the set of all edges between item variable nodes, and Z is a normalization factor.

2.2.2 Probabilistic Message-Passing

To efficiently infer the marginal distribution $P(r_{zi}|\mathbb{R})$, $\forall i \in \mathbb{I}\backslash I_z$, from $P(\{r_{zi}, i \in \mathbb{I}\backslash I_z\}|\mathbb{R})$, we employ the loopy BP message-passing algorithm where probabilistic messages are iteratively exchanged between item nodes on PMRF. In the n-th iteration, the message $m_{i,j}^{(n)}(r_{zj})$ from item i to its connected item j is given by

$$m_{i,j}^{(n)}(r_{zj}) \propto \sum_{r_{zi}} \psi_{ij}(r_{zi}, r_{zj})\phi_i(r_{zi}) \prod_{h \in \mathcal{N}_i \backslash j} m_{h,i}^{(n-1)}(r_{zi}), \tag{22}$$

where $\mathcal{N}_i$ denotes the set of neighbor item nodes connected to node i. The message in (22) represents how one item should be rated according to the items it is connected to in PMRF. The m-messages are iteratively exchanged between item nodes. During each iteration, the outgoing messages at each item node are updated using incoming messages from the last iteration. Finally, every item i calculates its predicted rating value as the expectation of the marginal probability distribution. We summarize the BP algorithm for rating predictions in Algorithm 2.

Algorithm 2 Belief Propagation Algorithm over PMRF

- Initialization. Initialize $m_{i,j}(r_{zj})$ as $m_{i,j}^{(0)}(r_{zj}) = \frac{1}{|\Upsilon|}$, and set iteration counter $n = 1$.
- Iterative message-passing until convergence.
  - (1) Update $m_{i,j}^{(n)}(r_{zj})$ at all item nodes using (22);
  - (2) $n = n + 1$. Repeat (1) until convergence.
- Compute marginal item rating probability:

$$P(r_{zi}) \propto \phi_i(r_{zi}) \prod_{j \in \mathcal{N}_i} m_{j,i}(r_{zi}).$$

- Predict item rating as the expectation of the marginal probabilities:

$$\bar{r}_{zi} = \sum_{r \in \Upsilon} rP(r_{zi} = r).$$

We further note that the beliefs are exact if the PMRF has no loops. However, the graphical model for recommender systems has cycles. Due to the statistical dependencies between the messages in this loopy case, the marginal probabilities of the variables in $\mathbb{R}_z$ are approximations rather than exact values. However, it is shown that BP often works well even in the loopy graphs [122].

For each active user, the complexity for computing user similarities between other users and the active user on a factor graph is linear in the number of total users [127]. Then to predict all item ratings on a PMRF using BP, the complexity in terms of multiplications is $\mathcal{O}(|\Upsilon|NK^2)$, where K is the item neighborhood size on PMRF, which is a fixed parameter independent of M and N. Thus, the complexity is linear in the total number of items, N. Further, the proposed algorithm converges quickly, on the average in 10 iterations.

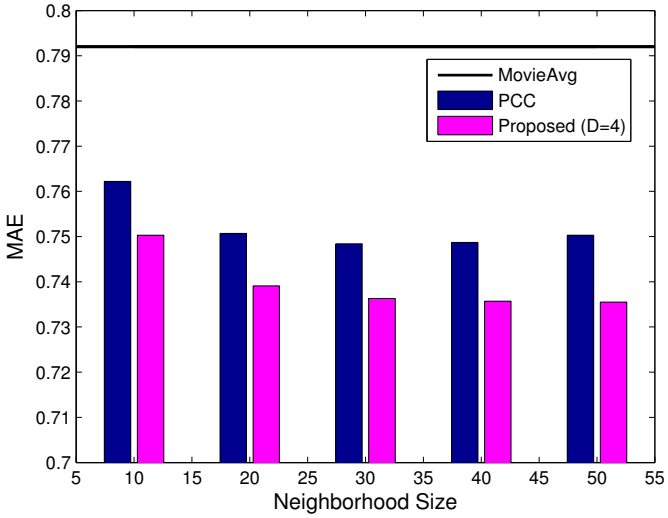
2.3 Experimental Evaluation

We evaluate the performance of the proposed algorithms using the 100K MovieLens dataset[1]. The dataset contains $100,000$ ratings, all integers from 1 to 5, on 1682 items (movies) by 943 users, and each user has rated at least 20 items. We randomly divide the users into two disjoint sets, 80% for training and 20% for testing. Specifically, for each user in the test set, we only keep a subset of its ratings as known (15 ratings in our setup), and use the rest of its ratings as test ratings.

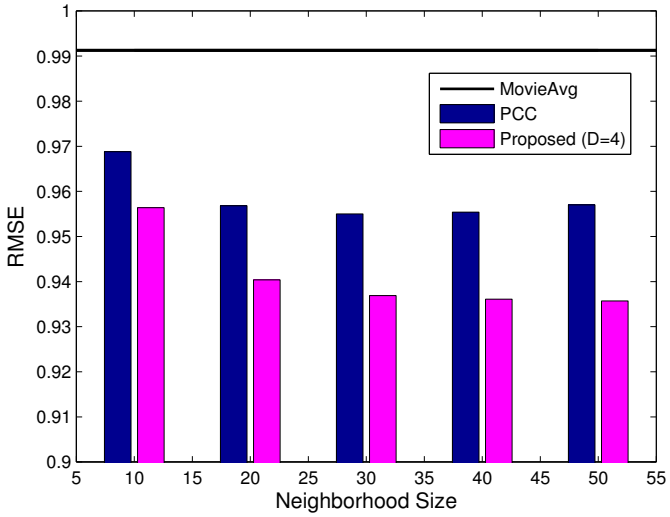
2.3.1 Comparison of Neighborhood Methods

We compare the prediction performance of the user-based neighborhood method when the user similarities are computed using our proposed algorithm in Sec. 2.1 and the PCC algorithm [99], in terms of both Mean Absolute Error (MAE) and Root Mean

[1]Available at: http://www.grouplens.org/node/73.



(a) MAE versus the effective neighborhood size.



(b) RMSE versus the effective neighborhood size.

Figure 7: Rating prediction performance comparison of neighborhood methods.

Squared Error (RMSE) metrics:

$$\text{MAE} = \frac{1}{|\mathbb{T}|} \sum_{r_{ui} \in \mathbb{T}} |r_{ui} - \hat{r}_{ui}|, \quad \text{RMSE} = \sqrt{\frac{1}{|\mathbb{T}|} \sum_{r_{ui} \in \mathbb{T}} (r_{ui} - \hat{r}_{ui})^2},$$

where $\mathbb{T}$ is the set of all test ratings for users in the test dataset, r_{ui} is the actual value of the rating provided by user u on the item i in the test dataset, and $\hat{r}_{ui}$ is the predicted rating value. Smaller RMSE and MAE errors mean better prediction accuracy. Note that the RMSE metric is more sensitive to large errors than MAE.

Table 1: Performance of proposed algorithm with varying group size D when $\mathcal{S} = \{1, 2\}$.

Group size	MAE		RMSE	
	$K = 10$	$K = 30$	$K = 10$	$K = 30$
$D = 3$	0.7533	0.7373	0.9593	0.9380
$D = 4$	0.7487	0.7358	0.9548	0.9362
$D = 5$	0.7522	0.7370	0.9587	0.9379

The results for the proposed algorithm and the PCC algorithm are presented in Fig. 7, where for the proposed algorithm, the similarity values of s_{uv} are taken from $\mathcal{S} = \{1, 2\}$, σ in (8) is set as $\sigma = 0.5$, and the group size D (i.e., the maximum allowed degree of factor node) is set as $D = 4$. We also show the results of the MovieAvg algorithm, which simply predicts ratings as the average of past ratings of each item. It can be observed from Fig. 7 that our proposed algorithm outperforms PCC in both MAE and RMSE under different neighborhood sizes. When the neighborhood size increases from 10 to 30 the performances of both algorithms improve, but the gain from increasing neighborhood size quickly diminishes and finally the performance even starts to degrade when reaching 50. This is because as neighborhood size increases, ratings from users with smaller similarity to the active user are also included to predict the rating, which corrupts the prediction from other users with higher similarity.

We also investigate the impacts of different parameters on the proposed algorithm. In Table 1, we show the results of the proposed algorithm for varying group size D, where we set $\mathcal{S} = \{1, 2\}$, and $\sigma = 0.5$. It can be seen that $D = 4$ achieves the best performance. It is also interesting to notice that the performances under different D's are close, meaning the algorithm is not sensitive to the choice of D. Since a large D dose not necessarily improve the performance and it exponentially increases computational complexity for generating λ-messages as discussed in Sec. 2.1.4, we should start from a small integer when searching for a good D for the algorithm.

In Table 2, we show the results for different $\mathcal{S}$, where $D = 3$ and $\sigma = 0.5$. We

Table 2: Performance of proposed algorithm with varying $\mathcal{S}$ when $D = 3$.

$\mathcal{S}$	MAE		RMSE	
	$K = 10$	$K = 30$	$K = 10$	$K = 30$
$\mathcal{S}_2$	0.7533	0.7373	0.9593	0.9380
$\mathcal{S}_5$	0.7524	0.7375	0.9584	0.9381
$\mathcal{S}_{10}$	0.7532	0.7379	0.9590	0.9386

Table 3: MAE performance comparison of PMRF and other PCC-based algorithms.

Algorithm	$K_{NE} = 10$	$K_{NE} = 15$
User-PCC	0.7622	0.7484
Item-PCC	0.7376	0.7420
Ensemble-PCC	0.7339	0.7310
PMRF	0.7261	0.7235

denote $\mathcal{S}_L = \{s : 1 \leq s \leq L\}$, where s only takes integer values and thus $|\mathcal{S}_L| = L$. It can be observed that the performances of the proposed algorithm for different $\mathcal{S}$'s are quite close, and $\mathcal{S}_5$ actually has slightly better performance than $\mathcal{S}_{10}$. This is because a large alphabet set $\mathcal{S}$ can causes overfitting, that is the computed user similarity is biased towards the training ratings and dose not generalize well on the test ratings.

2.3.2 Performance of PMRF Recommender System

We evaluate the prediction performance of the PMRF-based recommender system. We set the parameters as $s_{th} = 0.35$, and adjust the compatibility between two connected items using $\sigma_{ij} = \frac{1}{\sqrt{2}} \left(\frac{1-s_{ij}}{1-s_{th}} + 1 \right)$. The maximum number of similar items each item connected to is set as $K = 5$. The local evidence is computed according to (18), where the user similarities are computed on a factor graph as in Sec. 2.1 with $\mathcal{S} = \{1, 2, 3, 4, 5\}$, $D = 4$ and $\sigma = 0.5$.

We compare the proposed PMRF-based algorithm with other neighborhood algorithms using PCC [99] including the user-based PCC (User-PCC), the item-based

PCC (Item-PCC), and the ensemble of user-based PCC and item-based PCC algorithms (Ensemble-PCC) [67]. The MAE results are presented in Tab. 3, where K_{NE} denotes the neighborhood size. In the user-based method, K_{NE} refers to the effective number of similar users used, whereas in the item-based method, K_{NE} refers to the effective number of similar items used. In Ensemble-PCC, both the user-based and the item-based algorithms use the same K_{NE}, and their prediction outputs are combined. In PMRF, K_{NE} is the number of similar users used to compute local evidence in (18). The results show that the proposed PMRF algorithm outperforms other PCC-based neighborhood algorithms.

2.4 Conclusion

In this chapter, we proposed probabilistic graphical models for collaborative filtering recommender systems. Firstly, we proposed to compute user similarity on factors graphs for the neighborhood CF method. We introduced a proper factorization of the joint distribution function of similarity variables, which was expressed by an appropriate chosen factor graph, and take advantage of BP to efficiently compute the marginal distributions from their joint distribution. Since the resulting factor graph has loops, the "loopy" BP algorithm using iterative message passing was applied. We also proposed a complexity-reduction technique to contain the exponential increase in computational complexity due to the high degree at the factor node. The experimental results on 100K MovieLens dataset showed that the proposed similarity computation algorithm for the user-based neighborhood method achieves improved accuracy over the popular PCC algorithm in terms of both MAE and RMSE. Meanwhile, the computational complexity of the BP-based algorithm is comparable to that of PCC, growing linear in the number of users.

Secondly, we solved the recommender system problem on a Pairwise Markov Random Field (PMRF) incorporating both user similarity and item similarity. The BP

algorithm was utilized to efficiently estimate the marginal probability distributions of the item ratings via iterative message-passing between item nodes. The experimental results show that the proposed PMRF algorithm outperforms other PCC-based neighborhood algorithms. The complexity of the proposed algorithm remains linear per active user, making it very attractive for large-scale systems.

CHAPTER III

RECOMMENDATION VIA IMPLICIT PREFERENCE PROPAGATION IN SOCIAL NETWORKS

3.1 Introduction and Related Works

In Collaborative Filtering based recommender systems, the active user will be recommended items favorably rated by other users with similar tastes to the active user. The user similarity can be estimated based on user profiles including detailed personal information, but due to privacy concerns they are very difficult to obtain. Hence, most collaborative recommender systems evaluate user similarity based on users' historic rating data. This causes the cold-start problem for new users or users who do not provide enough ratings, i.e., the recommender systems cannot find similar users for them.

Recently, with the thriving of online social networks (OSN), the social collaborative recommendation has attracted significant attention. In social networks, people are more likely to connect to other people sharing similar interests, and they are influenced more by people they connect to, further fostering similarity to each other [113]. Hence, by exploiting the social structure of social networks, social recommender systems can make satisfactory recommendations even for cold-start users when provided with their social connections. Moreover, social recommendation can also be conveniently incorporated with traditional collaborative filtering algorithms, improving their recommendation performance, especially for cold-start users. In [69, 68, 50, 120], social network was incorporated into matrix factorization methods, where the user latent factors are learnt with social relations taken into account. All of them require users' explicit rating data.

Today, people are increasingly concerned with their online privacy, adding new challenges to the recommendation problem. Although rating data do not directly tell personal details, it is still possible to infer user demographics, such as age and gender, from their ratings [114], and even uncover user identities and reveal sensitive personal information with access to other databases [81]. Therefore, users are not willing to make their personal data accessible to the general public. Hence, when designing collaborative recommendation systems, we need to take into account user privacy. Several works proposed to obfuscate user ratings with random noise, e.g., perturbation [91] and differential privacy [72], or to disguise genuine user profiles by adding extra fake data [104, 114]. However, such obfuscation techniques do not completely prevent rating data leakage. Alternatively, recommender systems can utilize only implicit data, e.g., whether a user has consumed an item or not, avoiding exposing explicit user rating data.

In this chapter, we propose a social recommendation algorithm that exploits the social network to generate recommendations using implicit user data [132]. We develop a probabilistic Bayesian Network (BN) model for item consumption in the social network, and infer the probability for each item to be selected by the active user given the observed implicit user data. Then, the top-N items with the highest probabilities are listed in the recommendation. Specifically, we propose an Expectation Propagation (EP) message-passing algorithm for approximate inference in the BN, based on which we further develop an iterative EP algorithm that performs EP message-passing for all users in a unified bipartite graph representation. We note that the original algorithm is a central scheme, in which the user data are collected and processed by a central authority. However, the algorithm can be easily adapted for distributed implementation, where users only exchange messages with friends they are directly connected to in social networks. This helps further protect user privacy, since users only share data with their immediate friends.

Previously, [49, 120] proposed social recommendation algorithms that can be combined with traditional neighborhood methods. In [49] the authors proposed a random walk approach TrustWalker, where the recommendation algorithm carries out random walk over the social network to collect ratings from other users, and computes the average ratings to generate recommendations for the active user. Obviously, user ratings are revealed to all users connected to the social network. [120] proposed a voting-based algorithm PureTrust, using only implicit user data. The algorithm collects votes from users found via Breadth First Search (BFS) in the social network. In contrast to [120], in our work we avoid direct information exchange between indirectly connected friends. Further, if a user has not consumed an item, instead of simply collecting a '0' vote for that item, our work assigns a random binary variable to model the willingness of the user to select that item, whose probability distribution depends on his directly connected friends.

The Bayesian networks were applied to recommender systems in [19, 119]. In [19], a Bayesian network is employed to model probabilistic item-based rating prediction, where each node corresponds to an item and its parent nodes correspond to similar items. In [119], the Bayesian network is also used to predict user ratings, but each node corresponds to an user in the social network. Different from these approaches, our work is interested in inferring the probabilities for items to be selected by users using only implicit data.

3.2 Top-N Recommendation Using Social Networks

3.2.1 Problem Description

We assume a set of M users denoted by $\mathbb{U} = \{1, \ldots, M\}$ and a set of L items denoted by $\mathbb{I} = \{1, \ldots, L\}$ in the social recommender system. Let U_i denote the set of users who have consumed item i, and I_u denote the set of items consumed by user u. Note that user u may also provide explicit feedback on item $i \in I_u$ in the form of rating

r_{ui}, but users keep such explicit data private. We arrange the collection of observed implicit data in a $M \times N$ matrix $\mathbb{S}$, where for each user u we place a '1' at the intersection of the u-th row and i-th column if $i \in I_u$ and leave the rest of the entries unfilled. Further, we assume that there is a social network established among users, where each user u is directly connected to a set of friends denoted by T_u. Let w_{uv} denote the trust value user u assigns to user $v \in T_u$, where larger trust value means higher degree of trust. w_{uv} can be some continuous value, e.g., $w_{uv} \in [0, 1]$, or can be binary value, where '1' indicates the existence of trust relations and '0' indicates unknown or no trust relations. Note that the trust relations are generally directed and asymmetric. We can represent the social network in a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices representing users in $\mathbb{U}$, and $\mathcal{E}$ is the set of edges representing social relations between users. In Fig. 8 we illustrate the graphical representation of a social network, where the weight on the edge is the trust value.

Given an active user z, the top-N recommendation task is to generate a list of N items from $\mathbb{I} \backslash I_z$ for user z. In this work, we focus on exploiting the social relations for collaborative recommendation. We construct probabilistic Bayesian networks to model the item consumption in the social network, and infer the probabilities for items to be selected by the active user given the observed data in $\mathbb{S}$. Then the top-N items with the highest probabilities are recommended to the active user.

3.2.2 Social Network-Based Bayesian Network

We define a binary variable s_{ui} to represent whether or not item i is selected by user u, $s_{ui} = 1$ if selected and $s_{ui} = 0$ otherwise. Hence, $s_{ui} = 1$ for item $i \in I_u$. However, for other items $i \in \mathbb{I} \backslash I_u$, instead of simply setting $s_{ui} = 0$, we consider them as unknown, since our goal is to recommend items from $\mathbb{I} \backslash I_u$ to user u. We are interested in inferring the probability distribution of s_{ui}, denoted by $BEL(s_{ui})$, $\forall i \in \mathbb{I} \backslash I_u$, given the observed data in $\mathbb{S}$. We make such inference following the fact that a user is most

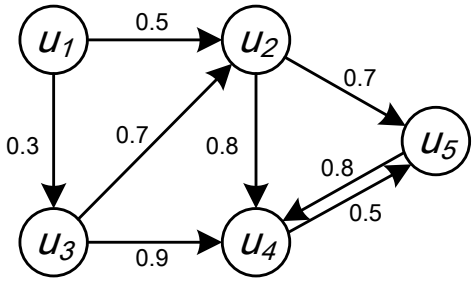


Figure 8: Graphical representation of the social network.

influenced by the friends he directly connect to in the social network.

We denote by $\mathcal{N}(u, K)$ a subset of K users with the highest trust values from user u's friend set T_u. To determine $\mathcal{N}(u, K)$ in scenarios where users do not explicitly specify trust values, e.g., assuming one for all trusted users, we first compute w_{uv}, $\forall v \in T_u$, as

$$w_{uv} = 1 + \frac{|I_u \cap I_v|}{|I_u \cap I_v| + c}, \tag{23}$$

where $c > 0$ is a constant. If user u has selected many items in common with user v in the past, user u is supposed to have high trust on user v. Given an active user z, to infer $BEL(s_{zi})$ for item i, we construct a Bayesian network $\mathcal{G}_{zi}$ as illustrated in Fig. 9. The root node at Layer 0 represents variable s_{zi}. Its parent nodes at Layer 1 are denoted by $\mathcal{P}_{zi} = \{s_{vi} : v \in \mathcal{N}(z, K)\}$, which correspond to the friends of user z. Similarly, for each node s_{vi} at Layer 1, we place its parent nodes at Layer 2. We repeat this process until Layer D. According to the conditional independence in the Bayesian network, each node s_{ui} is conditionally independent of other non-descendant nodes, given the configuration of its parent nodes $\mathcal{P}_{ui}$. We describe the conditional probability $P(s_{ui}|\mathcal{P}_{ui})$ as

$$P(s_{ui} = 1|\mathcal{P}_{ui}) = \frac{\sum_{v \in \mathcal{N}(u,K)} w_{uv} s_{ui}}{\sum_{v \in \mathcal{N}(u,K)} w_{uv}}, \tag{24}$$

where $P(s_{ui} = 0|\mathcal{P}_{ui}) = 1 - P(s_{ui} = 1|\mathcal{P}_{ui})$.

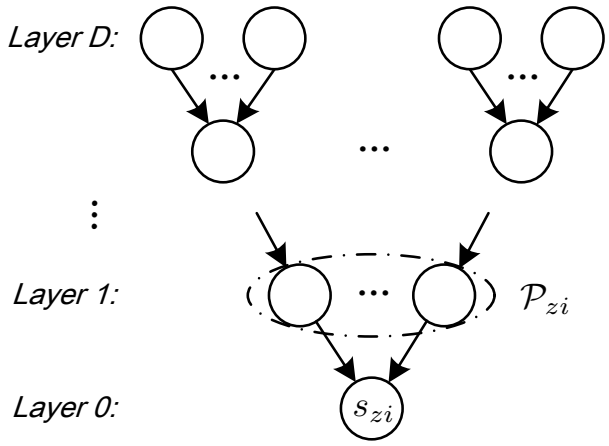


Figure 9: Bayesian network $\mathcal{G}_{zi}$ for active user z.

3.2.3 Inference and Expectation Propagation

3.2.3.1 Inference

Before introducing the inference process, we specify the observed and hidden variables in $\mathcal{G}_{zi}$. Let $\mathcal{V}_i = \{s_{vi} : v \in U_i\}$ be the set of observed variables for users who have consumed item i. Then $s = 1, \forall s \in \mathcal{V}_i$. Further, we assume the set $\mathcal{L}_{zi}(D)$ of variables at Layer D are also observed. Specifically, we set $s = 0$ for $s \in \mathcal{L}_{zi}(D) \cap \bar{\mathcal{V}}_i$, where $\bar{\mathcal{V}}_i = \{s_{vi} : v \in \mathbb{U} \backslash U_i\}$. We treat the rest of the variables in $\bar{\mathcal{V}}_i \backslash \mathcal{L}_{zi}(D)$ as hidden variables whose values are unknown.

We employ the Belief Propagation (BP) algorithm [87] to perform inference. However, exact inference can be intractable due to loops in the Bayesian network $\mathcal{G}_{zi}$. We instead resort to approximate inference by assuming a tree structure for $\mathcal{G}_{zi}$. Then inferring $BEL(s_{zi})$ is completed by passing probabilistic messages along the edges from the variable nodes at Layer d to the ones at Layer $d-1$, starting from Layer D. Specifically, we compute the message $\pi_a(s_{bi})$ sent to node s_{ai} at Layer $d-1$ from node $s_{bi} \in \mathcal{P}_{ai}$ at Layer d as follows

$$\pi_a(s_{bi}) = \alpha \sum_{\mathcal{P}_{bi}} P\left(s_{bi} | \mathcal{P}_{bi}\right) \prod_{q \in \mathcal{N}(b,K)} \pi_b(s_{qi}), \tag{25}$$

where α is a normalization factor such that $\sum_{s \in \{0,1\}} \pi_a(s_{bi} = s) = 1$. Note that if s_{bi}

is observed as $s_{bi} = \hat{s}_{bi}$, $\hat{s}_{bi} \in \{0, 1\}$, then $\pi_a(s_{bi} = \hat{s}_{bi}) = 1$ and $\pi_a(s_{bi})$ will not be updated. $\pi_a(s_{bi})$ expresses the probability distribution of s_{bi}, given the new evidence acquired by s_{bi} from incoming messages. Each node s_{ui} in $\mathcal{G}_{zi}$ generates messages sent to its children nodes once all messages from $\mathcal{P}_{ui}$ have arrived at s_{ui}.

The computational complexity in (25) is $\mathcal{O}\left(K2^K\right)$, which grows exponential with K. However, (25) can be simplified as following. Substituting (24) into (25), we have

$$\begin{aligned}\pi_a(s_{bi} = 1) &= \alpha \frac{\sum_{q \in \mathcal{N}(b,K)} \sum_{\mathcal{P}_{bi}} w_{bq} s_{qi} \prod_{q \in \mathcal{N}(b,K)} \pi_b(s_{qi})}{\sum_{q \in \mathcal{N}(b,K)} w_{bq}} \\ &= \alpha \sum_{q \in \mathcal{N}(b,K)} \hat{w}_{bq} \mathbb{E}_{\pi_b}(s_{qi}), \end{aligned} \tag{26}$$

where we define $\hat{w}_{bq} \triangleq \frac{w_{bq}}{\sum_{y \in \mathcal{N}(b,K)} w_{by}}$, and denote $\mathbb{E}(\cdot)$ as expectation operation. Note that $\mathbb{E}_{\pi_b}(s_{qi})$ is carried out with respect to the distribution $\pi_b(s_{qi})$. By using (26), the computational complexity of generating a single message is reduced to $\mathcal{O}(K)$. Moreover, since $\mathbb{E}_{\pi_a}(s_{bi}) = \pi_a(s_{bi} = 1)$, we can rewrite (26) as

$$\mathbb{E}_{\pi_a}(s_{bi}) = \alpha \sum_{q \in \mathcal{N}(b,K)} \hat{w}_{bq} \mathbb{E}_{\pi_b}(s_{qi}). \tag{27}$$

We hereafter refer to message passing using (27) as Expectation Propagation (EP). Finally, when all messages from nodes $\mathcal{P}_{zi}$ have arrived at s_{zi}, we compute $BEL(s_{zi})$ as

$$BEL(s_{zi} = 1) = \sum_{v \in \mathcal{N}(z,K)} \hat{w}_{zv} \mathbb{E}_{\pi_z}(s_{vi}). \tag{28}$$

We would like to clarify that, the propagation of messages between nodes is performed in a central server that collects and processes user data. Hence, there is no actual exchanges of messages between users in real world. However, the algorithm can be readily adapted for distributed implementation as explained in Sec. 3.2.3.3.

3.2.3.2 Iterative Expectation Propagation

In Secs. 3.2.2 and 3.2.3.1, we have shown that to infer $BEL(s_{zi})$ for an active user z, we need to construct a Bayesian network $\mathcal{G}_{zi}$ consisting of D layers of variable

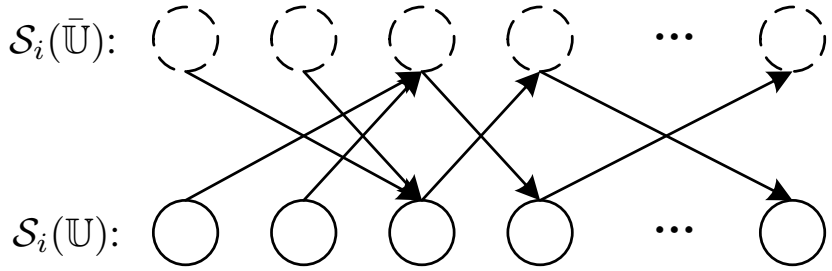


Figure 10: Iterative EP in a unified bipartite graph $\mathcal{G}_i$.

nodes. However, the intermediate results during the inference process in $\mathcal{G}_{zi}$ are not fully utilized. Indeed, there are significant amounts of repeated computations when performing inference for different users. In the following, we propose an iterative EP algorithm to simultaneously compute $\{BEL(s_{ui}) : u \in \mathbb{U}\}$ of all users for each item i in a single unified bipartite graph $\mathcal{G}_i$ that concisely encodes all social connections.

We create a set of virtual users, denoted by $\bar{\mathbb{U}} = \{\bar{1}, \ldots, \bar{M}\}$, where each virtual user $\bar{u} \in \bar{\mathbb{U}}$ is an image of a unique user $u \in \mathbb{U}$. We also assign a variable $s_{\bar{u}i}$ for each user $\bar{u}$. We represent the variables in $\mathcal{S}_i(\mathbb{U}) = \{s_{ui} : u \in \mathbb{U}\}$ and $\mathcal{S}_i(\bar{\mathbb{U}}) = \{s_{\bar{u}i} : \bar{u} \in \bar{\mathbb{U}}\}$ as two rows of nodes as illustrated in Fig. 10. Let $\bar{\mathcal{N}}(u, K)$ denote the set of images of users in $\mathcal{N}(u, K)$. For each variable node $s_{ui} \in \mathcal{S}_i(\mathbb{U})$, we connect to it the image nodes $\bar{\mathcal{P}}_{ui} = \{s_{\bar{v}i} : \bar{v} \in \bar{\mathcal{N}}(u, K)\}$ rather than its true parent nodes $\mathcal{P}_{ui}$. Similarly, we connect each $s_{\bar{u}i} \in \mathcal{S}_i(\bar{\mathbb{U}})$ to $\mathcal{P}_{ui}$ instead of $\bar{\mathcal{P}}_{ui}$. Then all layers of all concerned Bayesian networks are included in this unified representation $\mathcal{G}_i$. To see this, we can consider $\mathcal{S}_i(\bar{\mathbb{U}})$ and $\mathcal{S}_i(\mathbb{U})$ respectively as Layer $d+1$ and Layer d, when messages are sent from $\mathcal{S}_i(\bar{\mathbb{U}})$ to $\mathcal{S}_i(\mathbb{U})$ along the directed edges, and as Layer $d-1$ and Layer d, when messages are sent to $\mathcal{S}_i(\bar{\mathbb{U}})$ from $\mathcal{S}_i(\mathbb{U})$. Hence, the EP message-passing in any multi-layer Bayesian network can be carried out in $\mathcal{G}_i$ by iteratively passing messages between $\mathcal{S}_i(\bar{\mathbb{U}})$ and $\mathcal{S}_i(\mathbb{U})$. Moreover, we can simultaneously infer all $\{BEL(s_{ui}) : u \in \mathbb{U}\}$ in $\mathcal{G}_i$, reusing intermediate results and thus reducing overall computational complexity.

Following (27), the messages exchanged between variable nodes $\mathcal{S}_i(\bar{\mathbb{U}})$ and $\mathcal{S}_i(\mathbb{U})$

are given as follows

$$\mathbb{E}_{\pi_{\bar{u}}}(s_{vi}) = \sum_{\bar{q}\in\bar{\mathcal{N}}(v,K)} \hat{w}_{v\bar{q}}\mathbb{E}_{\pi_v}(s_{\bar{q}i}), \quad \forall s_{vi} \in \mathcal{S}_i(\mathbb{U}), \tag{29}$$

$$\mathbb{E}_{\pi_u}(s_{\bar{v}i}) = \sum_{q\in\mathcal{N}(\bar{v},K)} \hat{w}_{\bar{v}q}\mathbb{E}_{\pi_{\bar{v}}}(s_{qi}), \quad \forall s_{\bar{v}i} \in \mathcal{S}_i(\bar{\mathbb{U}}). \tag{30}$$

Note that passing messages using (29) and (30) for a total of D times is equivalent to performing inference in D-Layer Bayesian networks shown in Fig. 9. We initialize the messages as in Sec. 3.2.3. Assuming it starts from (29), we set $\mathbb{E}_{\pi_u}(s_{\bar{v}i}) = 1$ if $v \in U_i$, and $\mathbb{E}_{\pi_u}(s_{\bar{v}i}) = 0$ otherwise. In addition, during iterations, we always set both $\mathbb{E}_{\pi_u}(s_{\bar{v}i})$ and $\mathbb{E}_{\pi_{\bar{u}}}(s_{vi})$ to one for $v \in U_i$.

The resulting iterative EP inference algorithm is executed for every item in $\mathbb{I}$ in order to generate top-N recommendations. After obtaining $\{BEL(s_{ui}) : u \in \mathbb{U}\}$, $\forall i \in \mathbb{I}$, for each user u, we rank the items in $\mathbb{I}\backslash I_u$ in descending order of $BEL(s_{ui} = 1)$, and recommend the top-N items to user u. We observe that both steps (29) and (30) have the computational complexity of $\mathcal{O}(K)$. Then, the overall complexity of inferring $\{BEL(s_{ui}) : u \in \mathbb{U}\}$, $\forall i \in \mathbb{I}$, is $\mathcal{O}(DLMK)$, when using the proposed iterative EP algorithm to perform inference for all users simultaneously in the unified bipartite graph with D steps of message passing for each item.

3.2.3.3 Distributed Implementation

The proposed EP message-passing algorithm is very suitable for distributed implementation. To acquire recommendations from the social network, an active user z sends request to his directly connected friends $\mathcal{N}(z, K)$ for their information about items. In addition, user z also sends a decremental counter C with initial value D to its friends. If a friend $v \in \mathcal{N}(z, K)$ has consumed item i, he will send the message $\mathbb{E}_{\pi_z}(s_{vi}) = 1$ back to the requester user z. Otherwise, user v first decreases the counter C by 1, and if C is still greater than 0, he sends a request to his directly connected friends $\mathcal{N}(v, K)$ for information on item i, along with the counter C, otherwise he

sends the message $\mathbb{E}_{\pi_z}(s_{vi}) = 0$ back to user z. In cases that user v does send request to his friends, each friend u in $\mathcal{N}(v, K)$ repeats the same procedure as user v undergoes. Suppose user u sends request to his friends. Then he waits for response messages from $\mathcal{N}(u, K)$. Upon receiving all needed messages, user u generates the message $\mathbb{E}_{\pi_v}(s_{ui})$ according to (27), which is then sent back to user v. After the requester user z receives all messages from $\mathcal{N}(z, K)$, he calculates $BEL(s_{ui})$ according to (28).

In the distributed implementation of message-passing, the users only exchange messages with their directly connected friends in the social network. There is no need for users to reveal their data to the public. Also, each user mixes the messages received from his friends to generate a new message. Hence, it is difficult for users to extract the original data of friends of his friends from the received messages. Therefore, user privacy is further protected.

3.3 Experimental Results

We evaluate the top-N recommendation performance of the proposed EP algorithm using the Epinions[1] dataset prepared by [70]. The dataset consists of 49,290 users and 139,738 items. A total of 664,824 ratings are given by users on items, and each rating is an integer between 1 and 5. The dataset also includes 487,181 directed trust statements with trust value one. We randomly select 20% users for testing, and the rest for training. As in [49, 120], we define cold start users as users who have less than 5 ratings. Of the testing users, 41.5% are cold start users (52.8% in the overall dataset). For each user u in the testing set $\mathbb{T}$, we withhold an item, denoted by W_u, which has the maximum rating among items rated by this user. The performance of top-N recommendation is measured by recall computed as

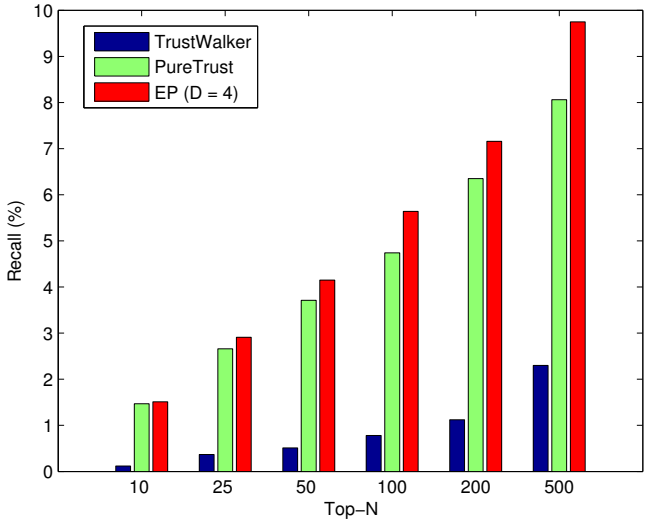
$$\text{Recall}(N) = \frac{\sum_{u \in \mathbb{T}} HIT(W_u, I_u(N))}{|\mathbb{T}|}, \tag{31}$$

[1] `http://www.trustlet.org/wiki/Epinions_datasets`.

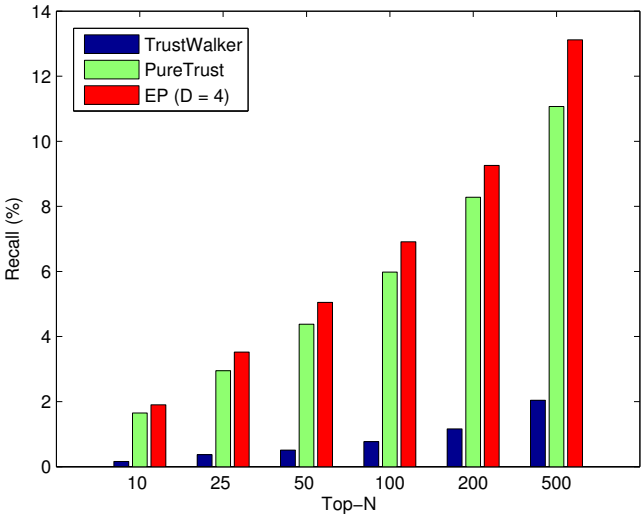
where $I_u(N)$ is the set of top-N items recommended to user u, and $HIT(\cdot) = 1$ if $W_u \in I_u(N)$ and $HIT(\cdot) = 0$ otherwise. Given N, higher recall means better performance.

We compare the performance of the proposed EP algorithm, the random walk based TrustWalker algorithm [49], and the voting-based PureTrust algorithm [120]. Note that due to the sparsity of the dataset, the recalls are quite low for all simulated recommendation algorithms when the number of recommended items is too small. In Fig. 11, we present the recall results for these algorithms for top-N recommendations when tested on (a) cold start users, and (b) all users. We set the neighborhood size K as 10 for both EP and PureTrust, and set D of EP as 4. We also set $c = 5$ in (23). The TrustWalker does not have a neighborhood size specification as it performs random walks over all possible users. The results indicate that the proposed EP outperforms the PureTrust algorithm. The achieved improvement for cold start users is 19% and 21% for top-100 and top-500 recommendations, respectively. The TrustWalker has the worst results as similarly observed in [120]. This is because many rating predictions generated by TrustWalker will have equal values, and the top-N items are selected without considering their popularity, which include many rarely rated items that have received one or two 5-ratings. As for computational complexity, to generate recommendations for all users, the proposed iterative EP requires $\mathcal{O}(DLMK)$ as discussed in Sec. 3.2.3.2, comparable to $\mathcal{O}(LMK)$ of PureTrust.

We further investigate the impacts of parameters K and D on the performance. To better illustrate the effects, we report results for top-500 recommendations, considering the sparsity of the dataset. In Fig. 12, we provide the top-500 recall results on cold start users under varying neighborhood size K. The results of EP are shown for both $D = 3$ and 4. It can be seen that EP achieves better results than PureTrust for all K from 10 to 100. In Fig. 13, we investigate the impact of D. We show the top-500 recall results on cold start users for D from 2 to 5, when $K = 10$ and 20.



(a) Cold start users



(b) All users

Figure 11: Comparison of recall results of top-N recommendations.

We observe that increasing D from 2 to 4 improves recall, but then as D increases further, the performance starts to saturate or even drop. This is because increasing D too high would allow the propagation of information from users that are further away from the active user in the trust network.

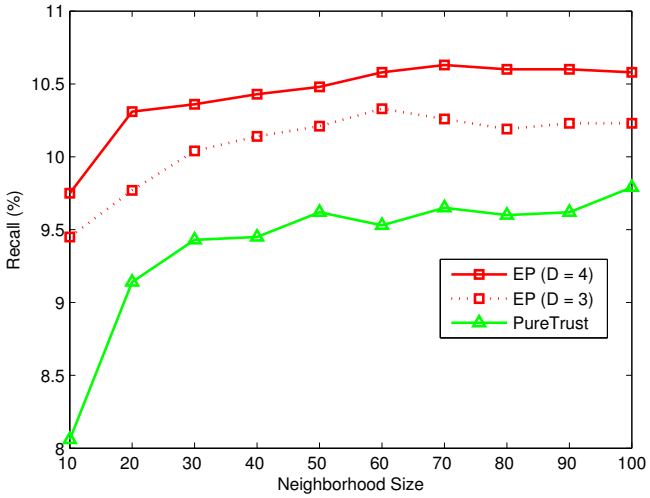


Figure 12: Top-500 recall versus neighborhood size.

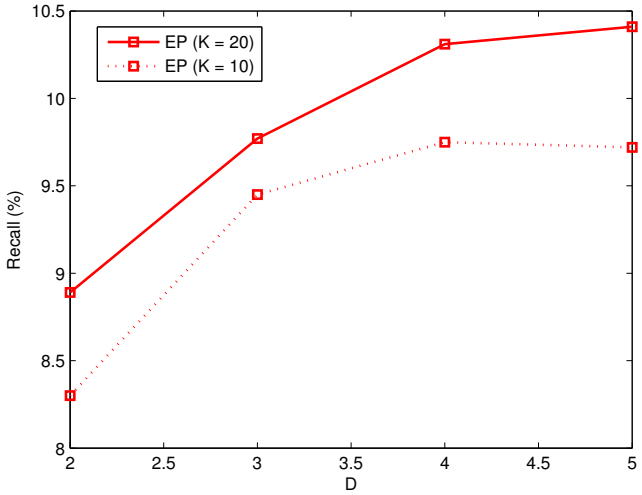


Figure 13: Top-500 recall versus D.

3.4 Conclusion

In this chapter, we proposed a Bayesian network based social recommendation algorithm that exploits the social relations between users in the social network to generate top-N recommendations, using only implicit user preference data. We constructed Bayesian networks based on the social network to model the probabilities for items to be selected by the active user, and developed the EP message-passing algorithm

to perform approximate inference efficiently. We also proposed an iterative EP algorithm to carry out EP message-passing for all users simultaneously in a unified bipartite graph representation. Moreover, the proposed message-passing algorithm is suitable for distributed implementation, where users only exchange messages with their directly connected friends. The experimental results on the Epinions dataset showed that the proposed EP algorithm achieves better top-N recommendation performance in terms of recall than both the random walk based and voting based social recommendation algorithms, while its computational complexity is comparable to theirs.

CHAPTER IV

PRIVACY-PRESERVING COLLABORATIVE FILTERING USING SEMI-DISTRIBUTED BELIEF PROPAGATION

4.1 Introduction

In traditional recommender systems, the recommendation algorithm is run at the central server of the commercial websites or other third party providers, and thus the users have to disclose their personal information, such as preferences, age and gender, to the server in order to receive satisfactory recommendation services. This raises the privacy issue, as users simply have no control over how their personal data will be disseminated and used [1]. It is not uncommon that websites sell to other parties such data, which are valuable for targeted advertising. As users become more concerned about their online privacy, they are less willing to directly release their personal information. Since detailed user profiles are difficult to obtain, most recommender systems employ the CF recommendation approach which only relies on the user ratings. However, it is still possible to infer from ratings user demographics, such as age and gender, from their ratings [114], and even uncover user identities and reveal sensitive personal information with access to other databases [81]. While personalized recommendations at e-commerce websites have been very successful [4], users are increasingly worried about online privacy [92]. Privacy-preserving CF recommender systems are thus in urgent need. The challenge stems from the conflict between accuracy and privacy. That is, to provide recommendations that better match the user's tastes, the system needs to know more about the user.

Most privacy-preserving recommender systems are developed either by obfuscating user data with random noise [91] or fake data [104] at the cost of recommendation

accuracy, or by encrypting user data using cryptography techniques [24] which require careful key management. In this work, we hope to build a recommender system with intrinsic privacy-preserving properties. To achieve this goal, it is evident that any communications between the server and users or between user peers should avoid exposing user ratings. An interesting work in [59] utilized concordance measure for computing user similarity required by the user-based CF algorithm, where computation is conducted between users without exposure of their ratings to each other, but unfortunately, to collaboratively generate recommendation, users need to reveal ratings to other users. We notice that user-based CF relies on direct collaboration among users, i.e., a user needs to know what other people like to find out what he might like, whereas item-based CF exploits the consistency in an individual user's taste, i.e., if a user likes an item, he might also like other items similar to it. We consider item-based CF a better choice for privacy-preserving recommender systems. The accuracy of the item-based CF system depends on the measure of item similarity, which has to be estimated based on ratings on items, and further, subject to the privacy constraint.

In this chapter, we introduce a semi-distributed Belief Propagation (BP) approach to privacy-preserving item-based CF recommender systems [128, 130]. Firstly, we formulate the item similarity computation, a key step of item-based CF, as a probabilistic inference problem on a proper factor graph, which can then be efficiently solved using BP. Then we develop a semi-distributed architecture for BP, where probabilistic messages on item similarity are exchanged between the server and users, without disclosing user ratings to the server or other peer users. We further propose a cascaded BP scheme to address the practical issue that only a subset of users are available for BP during one time slot. Each user locally generates rating predictions by averaging his own ratings on items weighted according to their similarities to other unseen items. Hence, the proposed item-based CF algorithm preserves user privacy in both

item similarity computation and rating prediction. We analyze the achieved privacy of the semi-distributed BP from a information-theoretic perspective. We also propose a method that reduces the computational complexity of BP at the user side. Through experiments on the MovieLens dataset, we show that our algorithm achieves superior accuracy.

4.1.1 Related Works

Recently, privacy-preserving information processing and data mining techniques have been widely studied. In [116], the authors proposed distributed online learning with intrinsic privacy-preserving properties, where local parameters at each participating user are updated based on local data sources, and parameters are periodically exchanged among a small subset of neighbors, and they showed that malicious users cannot reconstruct the subgradients of other users. In [97], the authors applied homomorphic encryption to privacy-preserving distributed aggregation of energy consumption metering data in smart grids. A number of works also investigated privacy-preserving information processing in semi-distributed or centralized cloud computing. [125] proposed privacy-preserving back-propagation neural network learning via cloud computing, where private data of each party are first encrypted before being uploaded to the cloud, and operations over ciphertexts are supported via doubly homomorphic encryption. [25] addressed multi-keyword ranked search over encrypted data in cloud computing using secure inner product computation.

There are existing works on both centralized and distributed privacy-preserving recommendation systems. The privacy-preserving techniques that have been applied to centralized systems mainly include data obfuscation and cryptography approaches. The data obfuscation approach is to obfuscate user ratings with random noise before releasing them to the server, e.g., differentially private recommender system [72] and

the perturbation technique [91]. In [104, 114], the authors proposed to disguise genuine user profiles by adding extra fake data. However, obfuscation and random noise undermine accuracy, and users have to sacrifice more privacy for better recommendation. In [102], the authors proposed a data perturbation approach based on differential privacy with accuracy guarantees.

The cryptography approach encrypts users data using advanced cryptography techniques to hide user information, e.g., item ratings, from the recommender server, which only operates on the encrypted user data [35, 34]. In particular, [34] proposed an item-based privacy-preserving recommender system with homomorphic encryption, where the server maintains an item-item similarity model, and generates rating predictions blindly by performing homomorphic operations on the encrypted ratings. However, the authors assumed the similarity model is known *a priori*, and did not provide any privacy-preserving solution for that. Besides, key management required by cryptography tools could be demanding in practice. [82] applied the garbled circuits cryptographic technique to matrix factorization collaborative filtering, where a third-party crypto-service provider is required for private computation. [13] discussed practical implementations of privacy-preserving collaborative filtering deployed in cloud computing infrastructures.

Alternatively, distributed privacy-preserving recommender systems let users store data in their devices. [76] presented a personal collaborative filtering recommender running on the user side. Each user stores his data locally, and constructs an item-item similarity model using the cosine similarity measure by incrementally incorporating ratings from other neighbour users in a peer-to-peer (P2P) environment. The quality of the locally constructed similarity model depends on the set of neighbours a user can find and contact. The best performance is achieved by using ratings from all common users of two items to evaluate the item similarity, which is much harder to implement in P2P architectures than in a centralized CF system. Moreover, the

user privacy is not guaranteed as users need to share their ratings with each other. An attacker can easily mimic the behaviour of genuine users to acquire their personal data. As in centralized systems, cryptography [24] and obfuscation techniques [14] can be applied to distributed systems to avoid disclosing data to other users. A client-side content-based mobile shopping recommender was proposed in [88], where user data, e.g., historical purchases, are locally stored on users' devices, e.g., smartphones, and each device runs its own content-based recommendation algorithm to find products that match the user's profile based on product descriptions. However, the complete product catalog can be challenging for mobile devices to process, and incurs significant data traffic as well. In [63], the authors proposed social connection recommendation in mobile social networking, where matching user attributes are computed via secure multi-party computation (SMC) techniques based on secret sharing.

4.1.2 Background on Item-based CF

Assuming there are M users and N items in a recommender system, let $\mathbb{U} = \{1, \ldots, M\}$ represent the set of all users, and $\mathbb{I} = \{1, \ldots, N\}$ represent the set of all items. A user u expresses his opinion on item i in the form of rating r_{ui} which takes values from a finite discrete set $\Gamma = \{r | 1 \leq r \leq T, r = 1, 2, \ldots\}$, e.g, $\Gamma = \{1, 2, 3, 4, 5\}$. We arrange the collection of all ratings in an incomplete $M \times N$ matrix $\mathbb{R}$, with r_{ui} at the intersection of u-th row and i-th column. The entries of unknown ratings are unfilled. Let I_u denote the subset of items rated by user u, and U_i denote the subset of users who have rated item i. The task of a recommender system is to predict the ratings for an active user u on the subset of unseen items $\mathbb{I} \backslash I_u$. Throughout this chapter, we focus on the item-based CF recommendation algorithm [99].

To predict the rating r_{ui} for user u on an unseen item i, the algorithm sorts the items in I_u according to their similarity to item i in descending order, and finds a subset of top K most similar items, denoted by $\mathcal{N}_{ui}$, where $|\mathcal{N}_{ui}| = K$. We refer to $\mathcal{N}_{ui}$

as the neighbourhood of item i from user u's perspective, and K as the neighborhood size. Then r_{ui} is predicted by

$$\hat{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_{ui}} s_{ij} \times r_{uj}}{\sum_{j \in \mathcal{N}_{ui}} |s_{ij}|}, \tag{32}$$

where s_{ij} is the similarity between items i and j. The accuracy of the algorithm depends on the measure of item similarity s_{ij}, which is computed based on the observed ratings on items. The user-based CF algorithm uses a similar formula to (32), but based on ratings from other users.

Several well-known methods for item similarity computation include Cosine Similarity (CS), Pearson Correlation Similarity (PCS), and Adjusted Cosine Similarity (ACS) [99]. Yet, all of those mentioned methods directly operate on ratings from different users, which requires users to disclose ratings to the server or other users. We will introduce a semi-distributed BP algorithm for item similarity computation with intrinsic privacy-preserving property.

4.2 Modelling Item Similarity on Factor Graphs

4.2.1 Probabilistic Problem Formulation

We model the similarity s_{ij} between items i and j as a discrete random variable that takes values from a predefined set $\mathcal{S}$. The total number of possible values is $L = |\mathcal{S}|$. Let $\mathbb{S}_i = \{s_{ij} : 1 \leq j \leq N, j \neq i\}$ be the set of item similarities between item i and other items. We denote by $P(\mathbb{S}_i|\mathbb{R})$ the joint posterior probability distribution of $\mathbb{S}_i$. To obtain s_{ij}, we need the marginal posterior probability distribution of s_{ij}, which can be derived by

$$P(s_{ij}|\mathbb{R}) = \sum_{s_{i1} \in \mathcal{S}} \cdots \sum_{s_{i(j-1)} \in \mathcal{S}} \sum_{s_{i(j+1)} \in \mathcal{S}} \cdots \sum_{s_{iN} \in \mathcal{S}} P(\mathbb{S}_i|\mathbb{R}). \tag{33}$$

For notational convenience, we rewrite (33) for the sum over all variables in $\mathbb{S}_i$ except s_{ij} as

$$P(s_{ij}|\mathbb{R}) = \sum_{\mathbb{S}_i \setminus s_{ij}} P(\mathbb{S}_i|\mathbb{R}). \tag{34}$$

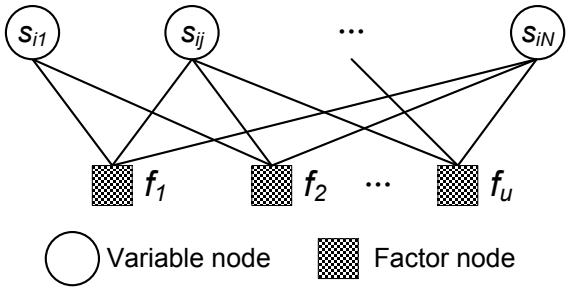


Figure 14: The factor graph $\mathcal{G}_i$ for item similarity.

However, direct computation using (34) is not appealing, because it must be performed by a central server, and thus users are required to disclose their ratings to the server. Moreover, (34) incurs an exponential complexity of $\mathcal{O}\big(L^N\big)$. We instead resort to the factor graph to express the factorization of $P\left(\mathbb{S}_i|\mathbb{R}\right)$, and apply the efficient sum-product BP algorithm to infer the marginal posterior probability distributions. More importantly, as will be described in Sec. 4.3, the BP algorithm can be carried out in a semi-distributed manner, so that the part of computation that requires knowledge of user ratings can be locally performed by the user, eliminating the need to disclosing user ratings.

4.2.2 Modelling with Factor Graphs

We first find a proper factorization for $P\left(\mathbb{S}_i|\mathbb{R}\right)$. For each user $u \in U_i$, we denote $\mathbb{S}_{ui} = \{s_{ij} : j \in I_u \backslash i\}$ as the set of item similarities between item i and other items user u has rated, and use a local function $f_u(\mathbb{S}_{ui})$ to model the dependencies among variables in $\mathbb{S}_{ui}$ from user u's perspective. Hence, $P\left(\mathbb{S}_i|\mathbb{R}\right)$ factorizes into local functions as follows

$$P\left(\mathbb{S}_i|\mathbb{R}\right) = \frac{1}{Z} \prod_{u \in U_i} f_u(\mathbb{S}_{ui}), \tag{35}$$

where Z is a normalization constant. We construct a factor graph $\mathcal{G}_i$ for the factorization in (35) as illustrated in Fig. 14, where there are $|\mathbb{S}_i|$ variable nodes and $|U_i|$ factor nodes. Each $s_{ij} \in \mathbb{S}_i$ is represented by variable node j in $\mathcal{G}_i$, and each local function $f_u(\mathbb{S}_{ui})$ is represented by factor node u. The subset of variable nodes for $\mathbb{S}_{ui}$

are connected to factor node u via edges. Let U_{ij} denote the set of common users of items i and j, $U_{ij} = U_i \cap U_j$. Hence, variable node j is connected to $|U_{ij}|$ factor nodes in $\mathcal{G}_i$. Essentially, if a user u has rated item i and other items in $I_u \backslash i$, then this user has a belief on the similarity s_{ij}, $\forall j \in I_u \backslash i$, from his perspective. The factor graph allows users' beliefs be exchanged and aggregated following the principle of sum-product message passing. To solve for all item similarities $\{\mathbb{S}_i : 1 \leq i \leq N\}$, a total of N such factor graphs need to be constructed.

The local function $f_u(\mathbb{S}_{ui})$ determines how user u estimates item similarity based on his own ratings. It should be properly designed with regard to the eventual goal to predict ratings using (32). For a user u who has rated item i, we assume for now rating r_{ui} is unknown, and let $\hat{I}_u = I_u \backslash i$. Given a configuration of item similarities in $\mathbb{S}_{ui}$, user u predicts r_{ui} as

$$\hat{r}_{ui}(\mathbb{S}_{ui}) = \frac{\sum_{j \in \hat{I}_u} s_{ij} \times r_{uj}}{\sum_{j \in \hat{I}_u} |s_{ij}|}. \tag{36}$$

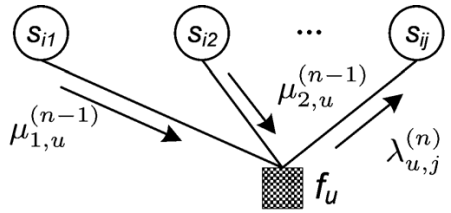
Note that (36) has a similar form to (32). Then user u checks $\hat{r}_{ui}(\mathbb{S}_{ui})$ against the actual rating r_{ui} using the following factor node function

$$f_u\left(\mathbb{S}_{ui}\right) = \frac{1}{Z_u} \exp\left\{-\frac{1}{\sigma^2}\left(\hat{r}_{ui}(\mathbb{S}_{ui}) - r_{ui}\right)^2\right\}, \tag{37}$$

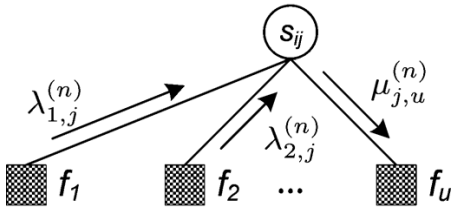
where Z_u is a normalization constant, and σ is a designing parameter that controls the sensitivity of $f_u\left(\mathbb{S}_{ui}\right)$ to the discrepancy between $\hat{r}_{ui}(\mathbb{S}_{ui})$ and r_{ui}. We note that $f_u\left(\mathbb{S}_{ui}\right)$ decreases with increasing discrepancy.

4.2.3 BP for Similarity Computation

We describe the sum-product BP algorithm to infer the marginal posterior distribution $P\left(s_{ij}|\mathbb{R}\right)$, $\forall s_{ij} \in \mathbb{S}_i$, on the factor graph $\mathcal{G}_i$, without worrying about privacy. Later in Sec. 4.3.1, we will introduce the semi-distributed implementation of BP for privacy, yet with no impacts on the computed similarities. Since the constructed factor graph has loops, we apply the loopy BP algorithm that iteratively exchanges



(a) The λ-message $\lambda_{u,j}^{(n)}(s_{ij})$.



(b) The μ-message $\mu_{j,u}^{(n)}$.

Figure 15: Illustration of message passing at iteration n.

messages between factor nodes and variable nodes along the edges until convergence [54]. As in the sum-product principle, there are two types of messages passed on $\mathcal{G}_i$: (i) The λ-message $\lambda_{u,j}(s_{ij})$ sent from a factor node u to a variable node j, and (ii) the μ-message $\mu_{j,u}(s_{ij})$ sent from a variable node j to a factor node u.

We illustrate the message passing in Fig. 15. In each iteration, each node (factor node or variable node) in the factor graph generates and sends messages to the neighbor nodes connected to it, based on incoming messages. In iteration n, factor node u generates the λ-message $\lambda_{u,j}^{(n)}(s_{ij})$ sent to variable node j by computing the product of local factor function $f_u(\mathbf{s}_{ui})$ with all μ-messages received in the previous iteration from neighbor variable nodes of factor node u, excluding the message from the recipient variable node j, and sums out all variables except s_{ij} as follows

$$\lambda_{u,j}^{(n)}(s_{ij}) \propto \sum_{\mathbb{S}_{ui} \backslash s_{ij}} f_u(\mathbb{S}_{ui}) \prod_{h \in \hat{I}_u \backslash j} \mu_{h,u}^{(n-1)}(s_{ih}). \tag{38}$$

The λ-message $\lambda_{u,j}^{(n)}(s_{ij})$ is a list of the beliefs on the similarity $s_{ij} = s$, $\forall s \in \mathcal{S}$, perceived from user u's perspective, given the current collective knowledge of the similarity between other items in $\hat{I}_u \backslash j$ and item i.

Variable node j generates the μ-message $\mu_{j,u}^{(n)}(s_{ij})$ sent to factor node u as the product of all incoming λ-messages received in the current iteration from all factor nodes connected to variable node j, excluding the one from the recipient factor node u as follows

$$\mu_{j,u}^{(n)}(s_{ij}) \propto \prod_{f \in F_j \backslash u} \lambda_{f,j}^{(n)}(s_{ij}), \tag{39}$$

where F_j denotes the set of factor nodes connected to variable node j. Here in graph $\mathcal{G}_i$, $F_j = \{u : u \in U_{ij}\}$. The μ-message $\mu_{j,u}^{(n)}(s_{ij})$ is a list of beliefs on the similarity $s_{ij} = s$, $\forall s \in \mathcal{S}$, which is generated by aggregating beliefs from other users in $U_j \backslash u$ on the similarity s_{ij}.

After convergence, the marginal posterior distribution $P(s_{ij}|\mathbb{R})$ is computed at variable node j as product of all λ-messages received from neighbor factor nodes connected to variable node j as

$$P(s_{ij}|\mathbb{R}) = \frac{1}{Z_{ij}} \prod_{f \in F_j} \lambda_{f,j}^{(n)}(s_{ij}), \tag{40}$$

where Z_{ij} is a normalization constant. Finally, based on the marginal posterior distribution $P(s_{ij}|\mathbb{R})$, the item similarity s_{ij} can be estimated in various ways. We consider using the minimum mean squared error criterion, for which the optimal estimated s_{ij} is given by the expectation

$$\hat{s}_{ij} = \sum_{s \in \mathcal{S}} s \times P(s_{ij} = s|\mathbb{R}). \tag{41}$$

We summarize the BP algorithm for computing $\mathbb{S}_i$ on factor graph $\mathcal{G}_i$ in Alg. 3.

4.3 Semi-Distributed Privacy-Preserving CF

Our goal is to build a privacy-preserving recommender system. Specifically, we would like to preserve user privacy for both item similarity computation and recommendation generation. In the following, we introduce the semi-distributed architecture of the proposed BP algorithm for item similarity computation without exposing user ratings, and the user-side recommendation generation as well.

Algorithm 3 BP on $\mathcal{G}_i$ for computing item similarity $\mathbb{S}_i$.

- Initialize all messages as $\lambda_{u,j}^{(0)}(s_{ij} = s) = \frac{1}{L}$ and $\mu_{j,u}^{(0)}(s_{ij} = s) = \frac{1}{L}$, $\forall s \in \mathcal{S}$, and set iteration $n = 1$.
- Iterative message passing until convergence.
  (a) Update all λ-messages using (38);
  (b) Update all μ-messages using (39);
  (c) $n = n + 1$. If not convergent, repeat (a) and (b).
- Compute marginal posterior probability distributions of $s_{ij} \in \mathbb{S}_i$ using (40).
- Compute item similarity $s_{ij} \in \mathbb{S}_i$ using (41).

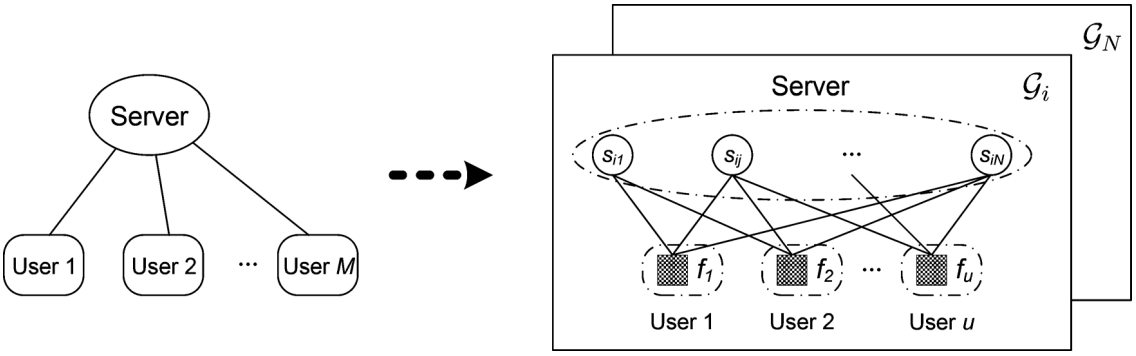


Figure 16: Architecture of semi-distributed BP.

4.3.1 Semi-Distributed BP

From Sec. 4.2.3, we know that the messages exchanged between factor nodes and variable nodes are probabilistic statements on item similarity, and such messages do not directly reveal user ratings. However, the user rating data are used to compute $f_u(\mathbb{S}_{ui})$ for generating λ-messages using (38). Hence, the key to preserve user privacy is to compute λ-messages at the user side. The μ-messages can be generated at a central server by performing multiplication operations on received λ-messages. This leads to the semi-distributed implementation of the BP algorithm. As we will see next, the computation follows the BP algorithm described in Sec. 4.2.3, and thus the semi-distributed BP does not impact the computed results.

Table 4: The format of λ-messages.

Graph i	User u	Item j	Iteration n	L-entry vector $\overrightarrow{\lambda}$

The architecture of the semi-distributed BP is shown in Fig. 16. The message passing on graph $\mathcal{G}_i$ is carried out by exchanging messages between the server and the users. Users locally store their personal ratings, and generate λ-messages according to (38) without disclosing their ratings to the server or other users. User u sends the λ-message $\lambda_{u,j}^{(n)}(s_{ij})$ to the server in the format shown in Table 4, where the L-entry vector $\overrightarrow{\lambda}$ stores the values of $\lambda_{u,j}^{(n)}(s_{ij} = s)$, $\forall s \in \mathcal{S}$. The server is responsible for generating μ-messages. To compute the μ-message $\mu_{j,u}^{(n)}(s_{ij})$, the server checks if all λ-messages $\lambda_{u,j}^{(n-1)}(s_{ij})$ from users in U_j have been received. The server then performs multiplication operations on the λ-messages to generate μ-messages according to (39), and sends the μ-message $\mu_{j,u}^{(n)}(s_{ij})$ to user u in the format shown in Table 5, where the L-entry vector $\overrightarrow{\mu}$ stores the values of $\mu_{j,u}^{(n)}(s_{ij} = s)$, $\forall s \in \mathcal{S}$. The server checks the convergence of messages, and obtains the item similarity after convergence using (41). Although we have applied synchronous BP here, asynchronous BP can be readily used, where messages are updated whenever new messages arrive. Indeed, round-robin asynchronous BP converges at least as fast as synchronous BP [33].

Table 5: The format of μ-messages.

Graph i	User u	Item j	Iteration n	L-entry vector $\overrightarrow{\mu}$

The computation of all item similarities in $\{\mathbb{S}_i : 1 \leq i \leq N\}$ requires N factor graphs, with $\mathbb{S}_i$ computed using factor graph $\mathcal{G}_i$. We introduce three protocols for coordinating the message passing on different graphs: the serial protocol, the parallel protocol, and the pipeline protocol. In the serial protocol, the message passing on factor graphs is performed in a serial manner, that is at any time, all generated messages belong to one factor graph, and unless inference on that graph is finished,

no messages on other factor graphs are generated. Alternatively, we can adopt the parallel protocol. If the bandwidth of the communication channel between users and the server is sufficient, and if the user's computational capability allows, message passing on multiple factor graphs can be performed in parallel to accelerate the inference process. The pipeline protocol, which is a combination of the serial and parallel protocols, is favorable when the delay in the network is large. While waiting for the messages on one factor graph to arrive, users can compute messages on other factor graphs and continue to send them to the server, so as to increase throughput and make more efficient use of computational resources as well.

Aside from the semi-distributed implementation, a fully distributed implementation is also possible. For example, [10] proposed a distributed BP-based trust management algorithm for P2P networks. Users directly send λ-messages to other users instead of to the server, and also generate μ-messages locally using received λ-messages, without relying on the server. However, there is a significant increase in communication overhead, considering that a λ-message $\lambda_{u,j}(s_{ij})$ needs to be sent to each of the users in U_j, as they require this λ-message for updating μ-messages. Moreover, for each item a user has rated, he needs to find out other users who also have rated that item, i.e., the graph must be known by the users. Further, since the item similarity is locally computed by users, and user $u \in U_i$ only obtains $\{s_{ij} : j \in \hat{I}_u\}$, the users need to share with each other the item similarities. Thus, a more sophisticated protocol needs to be developed for a fully distributed system.

4.3.2 Cascaded BP

The semi-distributed BP architecture in Sec. 4.3.1 requires that all users be active and participate in BP message propagation at the same time. Yet, in practical scenarios, it is difficult to meet this stringent requirement due to various reasons, e.g., some users may not be active temporally. To address this challenge, we adapt the original

BP architecture to perform BP in cascade. Rather than wait until all users become active, we can perform BP on the subgraph constructed based only on the subset of users active during the current time slot, and store the inferred knowledge for use in the next time slot. Let $\mathbb{U}^{(1)}, \mathbb{U}^{(2)}, \ldots, \mathbb{U}^{(t)}$ denote the available subsets of users at time slots $1, 2, \ldots, t$, respectively. We construct $\mathcal{G}_i^{(t)}$ based on $\mathbb{U}^{(t)}$ at each time slot t, and perform inference in $\mathcal{G}_i^{(t)}$. To incorporate the knowledge from the previous graph $\mathcal{G}_i^{(t-1)}$ into $\mathcal{G}_i^{(t)}$ at time slot t, we introduce the priors $P(s_{ij}|\mathcal{G}_i^{(t-1)})$, $\forall s_{ij} \in \mathbb{S}_i$, where $P(s_{ij}|\mathcal{G}_i^{(t-1)})$ is the marginal distribution of s_{ij} computed from $\mathcal{G}_i^{(t-1)}$. With these priors, the server computes the μ-messages according to

$$\mu_{j,u}^{(n)}(s_{ij}) \propto \prod_{f \in F_j \setminus u} \lambda_{f,j}^{(n)}(s_{ij}) \times P(s_{ij}|\mathcal{G}_i^{(t-1)}). \tag{42}$$

Upon convergence, the server computes the marginal distribution $P(s_{ij}|\mathcal{G}_i^{(t)})$ as follows

$$P\left(s_{ij}|\mathcal{G}_i^{(t)}\right) = \frac{1}{Z_{ij}} \prod_{f \in F_j} \lambda_{f,j}^{(n)}(s_{ij}) \times P(s_{ij}|\mathcal{G}_i^{(t-1)}). \tag{43}$$

Note that $P(s_{ij}|\mathcal{G}_i^{(t-1)})$ is actually computed by the server in the previous graph $\mathcal{G}_i^{(t-1)}$, and hence it is directly available for the server. The computation of λ-messages at the user side is still the same as (38). Hence, during cascaded BP, the priors are always directly computed and used at the server side.

The intermediate item similarity s_{ij} at time slot t can be computed as

$$\hat{s}_{ij} = \sum_{s \in \mathcal{S}} s \times P\left(s_{ij} = s|\mathcal{G}_i^{(t)}\right). \tag{44}$$

We will also update s_{ij} at each time slot of the cascaded BP. This allows us to generate recommendations using the most up-to-date item similarity.

4.3.3 User-Side Recommendation

Thus far, we have focused on the item similarity computation using the semi-distributed BP algorithm. To complete the privacy-preserving item-based CF recommender system, it remains to specify the recommendation generation. As introduced in Sec. 4.1.2,

Table 6: Summarization of entity functions.

Entity	Function
Server	Coordinate message passing; Generate μ-messages; Compute and store item similarity.
User	Store personal rating data; Generate λ-messages; Generate recommendations.

the item-based CF computes rating prediction for user u on item i using (32), which takes as its input the past ratings of user u and item similarities. To avoid revealing user ratings, users would then locally generate rating predictions. Since the item similarities are obtained at the server side in the semi-distributed BP algorithm, the server should send to users the required item similarities. To predict ratings on all unseen items in $\mathbb{I}\backslash I_u$, user u only needs item similarities in $\{s_{ij} : i \in \mathbb{I}\backslash I_u, j \in I_u\}$. After computing rating predictions, users can locally store the item similarities received from the server for future uses, and only update them periodically. We summarize the functions of the server and users in Table 6.

It is worth noting that in addition to preserving privacy, user-side recommendation also enhances user trust in e-commerce. Traditionally, centralized recommender systems owned by the commercial websites can manipulate the recommendations in various ways for revenue. A website might place the items with the highest profits on top of the recommendation list, or even employ recommendations as tools for advertisement of new products. This trust issue is well addressed by user-side recommendation, where users locally generate recommendations on their personal computers.

4.3.4 Accuracy and Communication Overhead

Assuming all users participate in BP, the semi-distributed architecture does not compromise the accuracy. Basically, we let users locally conduct the computations that

require private rating data, but all computations are carried out exactly as in a centralized approach in Sec. 4.2.3, and the user-side recommendation generates rating predictions using exactly (32). When not all users are available for BP at the same time slot, we can apply the cascaded BP scheme to improve performance over time. We will present performance evaluation later in Sec. 4.6.

The proposed algorithm incurs additional communication overhead. In the semi-distributed BP on graph $\mathcal{G}_i$, for each user $u \in U_i$, there are $|I_u|$ λ-messages and $|I_u|$ μ-messages exchanged between the server and use u in each iteration. And the server needs to send a total of $|I_u|(N - |I_u|)$ item similarities to user u for user-side recommendation. Whereas in the centralized approach, only the generated recommendations need to be sent to the user.

4.4 Information-Theoretic Privacy Analysis

We analyze the achieved privacy of the proposed algorithm for item-based CF by charactering the information leakage from the information-theoretic perspective. According to Sec. 4.3, the λ-messages and μ-messages are exchanged between the server and users, which can cause privacy degradation.

4.4.1 User Privacy Loss Due to λ-Messages

The λ-message $\lambda_{u,j}(s_{ij})$ in (38) is sent from user u to the server. Suppose the server tries to infer information from λ-messages about user u's ratings on items. We characterize the upper bound of information-theoretic privacy loss. To simplify the analysis, we assume s_{ij} can only take values from $\mathcal{S} = \{1, 2\}$. From (38) and (37), we can see that

$$\begin{cases} \lambda_{u,j}(s_{ij} = 1) < \lambda_{u,j}(s_{ij} = 2) & \text{if } |r_{uj} - r_{ui}| < \delta_{ui} \\ \lambda_{u,j}(s_{ij} = 1) \geq \lambda_{u,j}(s_{ij} = 2) & \text{if } |r_{uj} - r_{ui}| \geq \delta_{ui} \end{cases} \tag{45}$$

where $\delta_{ui} > 0$ is a parameter whose value is determined by ratings r_{uh} on other items $h \in \hat{I}_u \backslash j$ and the messages $\mu_{h,u}(s_{ih})$, $\forall h \in \hat{I}_u \backslash j$. Since the server does not have direct access to user ratings, δ_{ui} is unknown to it. To derive an upper bound, we assume that δ_{ui} is close to 1 with large probability, as the empirical results in [99] show that the item-based rating prediction method generates a rating with absolute error less than 1 on average using datasets with similar rating scale to Γ. Using this prior knowledge and (45), the server can infer about user ratings as follows

$$\begin{cases} r_{uj} = r_{ui} & \text{if } \lambda_{u,j}(s_{ij} = 1) < \lambda_{u,j}(s_{ij} = 2); \\ r_{uj} \neq r_{ui} & \text{if } \lambda_{u,j}(s_{ij} = 1) \geq \lambda_{u,j}(s_{ij} = 2). \end{cases} \tag{46}$$

We define the total privacy of user u, i.e., his rating information unknown to the server, as

$$\begin{aligned} H_u = H(\{r_{uk} | k \in I_u\}) &= \sum_{k \in I_u} H(r_{uk}) \\ &= |I_u| \log |\Gamma|, \end{aligned} \tag{47}$$

where $H(\cdot)$ denotes entropy. Note that we have assumed that user u rates different items independently, and the user rating on each item takes values from Γ with equal probabilities.

Next we compute the privacy loss $\Delta I(\lambda_{u,j}(s_{ij}))$ as the reduction of unknown information of user u's ratings after the server observes $\lambda_{u,j}(s_{ij})$

$$\begin{aligned} \Delta I(\lambda_{u,j}(s_{ij})) &= H(\{r_{uk} | k \in I_u\}) - H\Big(\{r_{uk} | k \in I_u\} \big| \lambda_{u,j}(s_{ij})\Big) \\ &= H(\{r_{uk} | k \in I_u\}) - \Big[H\Big(r_{uj} \big| \lambda_{u,j}(s_{ij}), \{r_{uk} | k \in I_u \backslash j\}\Big) \\ &\qquad + H\Big(\{r_{uk} | k \in I_u \backslash j\} \big| \lambda_{u,j}(s_{ij})\Big)\Big] \end{aligned} \tag{48}$$

Since $\lambda_{u,j}(s_{ij})$ only introduces dependency between r_{ui} and r_{uj}, we have

$$H\Big(r_{uj} \big| \lambda_{u,j}(s_{ij}), \{r_{uk} | k \in I_u \backslash j\}\Big) = H\left(r_{uj} | \lambda_{u,j}(s_{ij}), r_{ui}\right),$$

$$H\Big(\{r_{uk}|k \in I_u \backslash j\}\big|\lambda_{u,j}(s_{ij})\Big) = H(\{r_{uk}|k \in I_u \backslash j\}).$$

Hence,

$$\begin{aligned}\Delta I(\lambda_{u,j}(s_{ij})) &= H(\{r_{uk}|k \in I_u\}) - H\left(r_{uj}|\lambda_{u,j}(s_{ij}), r_{ui}\right) \\ &\quad - H(\{r_{uk}|k \in I_u \backslash j\}) \\ &= H(r_{uj}) - H(r_{uj}|\lambda_{u,j}(s_{ij}), r_{ui}). \end{aligned} \tag{49}$$

Specifically, for the two cases in (46), we have:

(1) If $\lambda_{u,j}(s_{ij}=1) < \lambda_{u,j}(s_{ij}=2)$

$$\begin{aligned}\Delta I_1(\lambda_{u,j}(s_{ij})) &= H(r_{uj}) - H(r_{uj}|r_{uj} = r_{ui}, r_{ui}) \\ &= \log|\Gamma|; \end{aligned} \tag{50}$$

(2) If $\lambda_{u,j}(s_{ij}=1) \geq \lambda_{u,j}(s_{ij}=2)$

$$\begin{aligned}\Delta I_2(\lambda_{u,j}(s_{ij})) &= H(r_{uj}) - H(r_{uj}|r_{uj} \neq r_{ui}, r_{ui}) \\ &= \log|\Gamma| - \log(|\Gamma|-1). \end{aligned} \tag{51}$$

The expected privacy loss per λ-message can be given by

$$\begin{aligned}\mathbb{E}\left\{\Delta I(\lambda_{u,j}(s_{ij}))\right\} &= \Delta I_1(\lambda_{u,j}(s_{ij}))P(r_{uj} = r_{ui}) \\ &\quad + \Delta I_2(\lambda_{u,j}(s_{ij}))P(r_{uj} \neq r_{ui}) \\ &= \log|\Gamma| - \frac{|\Gamma|-1}{|\Gamma|}\log(|\Gamma|-1), \end{aligned} \tag{52}$$

where $P(r_{uj} = r_{ui})$ is the probability that user u rates items i and j with the same rating, $P(r_{uj} = r_{ui}) = \frac{1}{|\Gamma|}$ and $P(r_{uj} \neq r_{ui}) = 1 - P(r_{uj} = r_{ui})$.

Taking into account all λ-messages from user u on graph $\mathcal{G}_i$, we can compute the

λ-message privacy loss for user u on graph $\mathcal{G}_i$ as

$$\begin{aligned}
&\Delta I\left(\{\lambda_{uj}(s_{ij})|j\in\hat{I}_u\}\right)\\
&=H\left(\{r_{uk}|k\in I_u\}\right)-H\left(\{r_{uk}|k\in I_u\}\big|\{\lambda_{u,j}(s_{ij})|j\in\hat{I}_u\}\right)\\
&=H\left(\{r_{uk}|k\in I_u\}\right)-H\left(\{r_{uk}|k\in I_u\backslash i\}\big|\{\lambda_{u,j}(s_{ij})|j\in\hat{I}_u\},r_{ui}\right)\\
&\quad-H\left(r_{ui}\big|\{\lambda_{u,j}(s_{ij})|j\in\hat{I}_u\}\right). \qquad (53)
\end{aligned}$$

Observing that

$$\begin{aligned}
&H\left(\{r_{uk}|k\in I_u\backslash i\}\big|\{\lambda_{u,j}(s_{ij})|j\in\hat{I}_u\},r_{ui}\right)\\
&\qquad=\sum_{j\in I_u\backslash i}H\left(r_{uj}\big|\lambda_{u,j}(s_{ij}),r_{ui}\right), \qquad (54)
\end{aligned}$$

$$H\left(r_{ui}\big|\{\lambda_{u,j}(s_{ij})|j\in\hat{I}_u\}\right)=H(r_{ui}), \qquad (55)$$

we have

$$\begin{aligned}
\Delta I\left(\{\lambda_{uj}(s_{ij})|j\in\hat{I}_u\}\right)&=\sum_{j\in\hat{I}_u}H(r_{uj})-\sum_{j\in\hat{I}_u}H(r_{uj}|\lambda_{u,j}(s_{ij}),r_{ui})\\
&=\sum_{j\in\hat{I}_u}\Delta I(\lambda_{u,j}(s_{ij})). \qquad (56)
\end{aligned}$$

Hence, the expected privacy loss resulting from all λ-messages from user u on graph $\mathcal{G}_i$ is

$$\begin{aligned}
\mathbb{E}\left\{\Delta I\left(\{\lambda_{u,j}(s_{ij})|j\in\hat{I}_u\}\right)\right\}&=\sum_{j\in\hat{I}_u}\mathbb{E}\left\{\Delta I(\lambda_{u,j}(s_{ij}))\right\}\\
&=|\hat{I}_u|\left(\log|\Gamma|-\frac{|\Gamma|-1}{|\Gamma|}\log(|\Gamma|-1)\right). \qquad (57)
\end{aligned}$$

As one example, suppose user u has rated 20 items with the rating scale $\Gamma=\{1,2,3,4,5\}$. His total privacy is $H_u=46.44$ bits, and the expected privacy loss due to λ-messages on one graph is 8.02 bits.

4.4.2 Total User Privacy Lost to the Server

In the previous analysis, we focused on the privacy loss due to messages on a single graph, but each user may participate in as many graphs as the number of items he rated, e.g., user u may send λ-messages on the set of graphs $\{\mathcal{G}_i | i \in I_u\}$. We are interested to know the total privacy loss of user u as a result of participating in multiple graphs. However, it is not simply the sum of privacy loss on all graphs, since the λ-messages on different graphs are not independent and thus there is significant redundancy in the revealed information. We next derive an upper bound of the total privacy loss.

Suppose the server is intelligent, and by combining the pairwise relationships between item ratings of user u inferred from λ-messages on multiple graphs using (46), it can divide the unknown ratings of user u in to $|\Gamma|$ groups, where the items in each group g have the same rating $r_g \in \Gamma$ while items in different groups have different ratings. Let G_g denote the set of items in group g, then $r_{uk} = r_g$, $\forall k \in G_g$. The unknown information of item ratings in each group g is

$$\begin{aligned} H(\{r_{uk} | k \in G_g\}) &= H\left(\{r_{uk} | k \in G_g \backslash i\} \left| r_{ui}\right.\right) + H(r_{ui}) \\ &= H(r_{ui}) \\ &= H(r_g). \end{aligned} \tag{58}$$

The unknown information of item ratings in all groups can be derived as

$$\begin{aligned} & H\left(\{r_{uk} | k \in G_1\}, \{r_{uk} | k \in G_2\}, \ldots, \left\{r_{uk} | k \in G_{|\Gamma|}\right\}\right) \\ &= H(r_1, r_2, \ldots, r_{|\Gamma|}) \\ &= H(r_1) + H(r_2 | r_1) + \ldots + H(r_{|\Gamma|} | r_1, r_2, \ldots, r_{|\Gamma|-1}) \\ &= \log |\Gamma| + \log(|\Gamma| - 1) + \ldots + \log 1 \\ &= \log(|\Gamma|!). \end{aligned} \tag{59}$$

Hence, the total privacy loss of user u is

$$\Delta I_u^{\text{total}} = |I_u| \log |\Gamma| - \log(|\Gamma|!). \tag{60}$$

In the example of 20 rated items with $|\Gamma| = 5$, the privacy loss is 39.53 bits versus the total privacy of 46.44 bits. However, without knowing exactly the rating of each group, the server will not be able to launch effective attacks against users to reveal sensitive personal information [114, 81]. Yet, the users are suggested to randomly participate in only a subset of graphs to better preserve their privacy.

4.4.3 User Privacy Lost to Other Users Due to μ-Messages

The μ-messages are sent from the server to users, and no direct messages are exchanged between users. Suppose a curious user u tries to extract from μ-messages information about ratings of other users. We notice that $\mu_{j,u}(s_{ij})$ in (39) mixes the λ-messages from multiple users, but does not contain any information about who are the source users that send those λ-messages. Thus, even if user u can extract any information from μ-messages, he does not know whom the information should be related to.

Nevertheless, we are interested in characterizing the amount of information a curious user can extract from μ-messages about ratings of other users, assuming the server colludes with some curious users and provides the graph structure information of $\mathcal{G}_i$ to them. We compute the upper bound of the rating information of other users a curious user u can infer about from the received μ-message $\mu_{j,u}(s_{ij})$, as well as the privacy of a particular user lost to user u.

From (39), we know that $\mu_{j,u}(s_{ij})$ is the product of λ-messages from multiple users in $U_{ij} \backslash u$. To simplify analysis, we consider the worst case of privacy loss where $\mu_{j,u}(s_{ij})$ is consistent with each λ-message $\lambda_{v,j}(s_{ij})$, i.e., user u can infer about λ-messages as follows

(1) If $\mu_{j,u}(s_{ij}=1) < \mu_{j,u}(s_{ij}=2)$:

$$\lambda_{v,j}(s_{ij}=1) < \lambda_{v,j}(s_{ij}=2);$$

(2) If $\mu_{j,u}(s_{ij}=1) \geq \mu_{j,u}(s_{ij}=2)$:

$$\lambda_{v,j}(s_{ij}=1) \geq \lambda_{v,j}(s_{ij}=2).$$

With the inferred λ-messages, user u can further infer about the ratings from other users in $U_{ij}\backslash u$. The inference process is the same as described in Sec. 4.4.1. Hence, the upper bound of the rating information of other users extracted from μ-message $\mu_{uj}(s_{ij})$ can be given by

$$\Delta I(\mu_{j,u}(s_{ij})) = \sum_{v \in U_{ij}\backslash u} \Delta I(\lambda_{v,j}(s_{ij})). \tag{61}$$

Using (52), we can obtain the expected amount of information extracted by user u from $\mu_{j,u}(s_{ij})$ as

$$\begin{aligned}\mathbb{E}\left\{\Delta I(\mu_{j,u}(s_{ij}))\right\} &= \sum_{v \in U_{ij}\backslash u} \mathbb{E}\left\{\Delta I(\lambda_{v,j}(s_{ij}))\right\} \\ &= (|U_{ij}|-1)\left(\log|\Gamma| - \frac{|\Gamma|-1}{|\Gamma|}\log(|\Gamma|-1)\right).\end{aligned} \tag{62}$$

The privacy of user v lost to a curious user u are essentially caused by the μ-messages on graph $\mathcal{G}_i$ that mix λ-messages from user v and are forwarded to user u by the server. Hence, we can derive the expected privacy of user v lost to user u on graph $\mathcal{G}_i$ as

$$\begin{aligned}&\mathbb{E}\left\{\Delta I\left(\{\lambda_{v,j}(s_{ij})|j \in \hat{I}_{uv}\}\right)\right\} \\ &= |\hat{I}_{uv}|\left(\log|\Gamma| - \frac{|\Gamma|-1}{|\Gamma|}\log(|\Gamma|-1)\right),\end{aligned} \tag{63}$$

where $\hat{I}_{uv} = \hat{I}_u \cap \hat{I}_v$ denotes the set of common items rated by users u and v.

4.5 Complexity Analysis and Reduction

The computational complexity of the BP algorithm is determined by the computation of λ-messages and μ-messages. While the complexity of generating a μ-message using (39) is $\mathcal{O}(|I_u|)$, the complexity of generating a λ-message using (38) is $\mathcal{O}\left(|I_u|L^{|I_u|}\right)$, where $L = |\mathcal{S}|$, which is exponential in the degree of the factor node, i.e., the number of items user u has rated. Unfortunately, in recommender systems, a user can rate over hundred of items. We thus propose a complexity reduction technique by controlling the degree of the factor node.

We randomly divide the variable nodes in $\hat{I}_u$ at factor node u into small groups of size D where D is a small integer. There are $G_u = \lceil |\hat{I}_u|/D \rceil$ such groups. Let $D_u^{(k)}$ denote the size of group k, $D_u^{(k)} = D$ for $1 \leq k \leq G_u - 1$ and $D_u^{(k)} = |\hat{I}_u| - D(G_u - 1)$ for $k = G_u$. Let $I_u^{(k)}$ denote the variable nodes in group k. We set an indicator $j_u^{(k)} = 1$ if $j \in I_u^{(k)}$ and $j_u^{(k)} = 0$ otherwise. Since each variable node $j \in \hat{I}_u$ only belongs to one group, we have $\sum_{k=1}^{G_u} j_u^{(k)} = 1$. Let $\mathbb{S}_{ui}^{(k)} = \{s_{ij} : j \in I_u^{(k)}\}$ denote the variable nodes in group k of $\hat{I}_u$. Assuming independence of variable nodes in different groups, we replace the factor node function $f_u(\mathbb{S}_{ui})$ in (37) with

$$\prod_{k=1}^{G_u} f_u^{(k)}\left(\mathbb{S}_{ui}^{(k)}\right), \tag{64}$$

where $f_u^{(k)}$ is the local function of variables in group k. Hence, we modify the factor graph $\mathcal{G}_i$ accordingly to obtain a new factor graph $\hat{\mathcal{G}}_i$ for complexity reduction. Instead of connecting all variable nodes in $\hat{I}_u$ to one factor node u, we connect a separate factor node $u^{(k)}$ to the group of variable nodes in $I_u^{(k)}$.

A more intuitive understanding is that, user u first randomly divides the items in $\hat{I}_u$ into multiple groups of size D, and decides item similarity on the basis of groups, as if items in different groups were rated independently by user u. Since variable node j is associated with item j, the variable nodes in $\hat{I}_u$ are grouped exactly as the grouping among items.

The local function at factor node $u^{(k)}$ can be similarly derived as (37). Assuming r_{ui} on item i is unknown, then using ratings from items within group k, user u predicts r_{ui} as

$$\hat{r}_{ui}(\mathbb{S}_{ui}^{(k)}) = \frac{\sum_{j \in I_u^{(k)}} s_{ij} \times r_{uj}}{\sum_{j \in I_u^{(k)}} |s_{ij}|}. \tag{65}$$

We substitute $\hat{r}_{ui}(\mathbb{S}_{ui})$ with $\hat{r}_{ui}(\mathbb{S}_{ui}^{(k)})$ in (37) to obtain the new local function of factor node $u^{(k)}$

$$f_u^{(k)}\left(\mathbb{S}_{ui}^{(k)}\right) = \frac{1}{Z_u^{(k)}} \exp\left\{-\frac{1}{\sigma^2}\left(\hat{r}_{ui}(\mathbb{S}_{ui}^{(k)}) - r_{ui}\right)^2\right\}, \tag{66}$$

where $Z_u^{(k)}$ is a normalization constant.

On the new factor graph $\hat{\mathcal{G}}_i$, we apply the BP algorithm described in Sec. 4.2.3. The λ-messages and μ-messages are exchanged between the new factor nodes and variable nodes. The λ-message $\lambda_{u^{(k)},j}^{(n)}(s_{ij})$ sent from factor node $u^{(k)}$ to variable node j is given by

$$\lambda_{u^{(k)},j}^{(n)}(s_{ij}) \propto \sum_{\mathbb{S}_{ui}^{(k)} \setminus s_{ij}} f_u(\mathbb{S}_{ui}^{(k)}) \prod_{h \in I_u^{(k)} \setminus j} \mu_{h,u^{(k)}}^{(n-1)}(s_{ih}). \tag{67}$$

And the μ-message $\mu_{j,u^{(k)}}^{(n)}(s_{ij})$ sent from variable node j to factor node $u^{(k)}$ is given by

$$\mu_{j,u^{(k)}}^{(n)}(s_{ij}) \propto \prod_{f \in \hat{F}_j \setminus u^{(k)}} \lambda_{f,j}^{(n)}(s_{ij}), \tag{68}$$

where $\hat{F}_j = \{v^{(k)} : v \in U_{ij},\, j_v^{(k)} = 1,\, 1 \le k \le G_v\}$.

The complexity of updating a λ-message is effectively reduced to $\mathcal{O}\left(DL^D\right)$ from $\mathcal{O}\left(|I_u|L^{|I_u|}\right)$ by using (67). Meanwhile, the total number of λ-messages need to be generated from user u's perspective in each iteration remains $|\hat{I}_u|$, which is the same as in $\mathcal{G}_i$. There is no change in complexity regarding the μ-messages. The overall complexity of the BP algorithm on $\hat{\mathcal{G}}_i$ with complexity reduction in each iteration is $\mathcal{O}\left(\bar{M}\bar{N}DL^D + N\bar{M}^2\right)$, where $\bar{N}$ is the average number of items rated by one user, and $\bar{M}$ is the average number of users of one item. Since the number of items a user can consume is limited by his time and money, we can assume $\bar{N}$ is much smaller than

N. As for $\bar{M}$, we assume $\bar{M}$ grows in the order of $M^{1-\epsilon}$, where $0 < \epsilon < 1$. Then we can rewrite the computation complexity on $\hat{\mathcal{G}}_i$ as $\mathcal{O}\left(M^{1-\epsilon}\bar{N}DL^D + NM^{2(1-\epsilon)}\right)$, and when N and M is large, it is dominated by the second term, so we have $\mathcal{O}\left(NM^{2(1-\epsilon)}\right)$.

The complexity reduction part can be easily integrated into the semi-distributed BP architecture in Sec. 4.3. Indeed, the complexity reduction is for reducing the computational burden of the users, and users can locally perform the required grouping. Also, the grouping information is not needed at the server side for BP. The server does not need to know from which group of user u the λ-message is generated, since only one $\lambda_{u,j}^{(n)}(s_{ij})$ message is generated by user u that is destined to variable node j on graph $\mathcal{G}_i$. Meanwhile, at the user side, user u can easily recover the group information "k" from the received message $\mu_{j,u}^{(n)}(s_{ij})$ by looking up for k with $j_u^{(k)} = 1$, and obtain $\mu_{j,u^{(k)}}^{(n)}(s_{ij})$, as if it was computed using (68), which can then be used to update $\lambda_{u^{(k)},j}^{(n)}(s_{ij})$ in (67).

Therefore, the grouping step is transparent to the server. The computational complexity on graph $\mathcal{G}_i$ at each user is only $\mathcal{O}\left(\bar{N}DL^D\right)$, regardless of N and M, and users can autonomously adjust D according to the availability of local computation resources and power.

4.6 Experimental Evaluation

We evaluate the accuracy of the proposed privacy-preserving item-based CF algorithm on the 100K MovieLens dataset[1], which consists of $100,000$ ratings on 1682 items (movies) by 943 users. Each rating is an integer between 1 and 5. We randomly divide the dataset into two disjoint sets: a training set containing 80% of the ratings, and a test set containing the rest 20% of the ratings. The ratings in the training set are used as memory for the item-based CF algorithm to compute item similarities and predict unknown ratings. We compare the predicted ratings with the actual ratings

[1]Available at: http://www.grouplens.org/node/73.

Table 7: MAE performance comparison of various item-based CF algorithms under different neighborhood size K with $N_c = 3$.

Algorithms	$N_c = 3$ ($P_{\text{pred}} = 74\%$)				
	$K = 10$	$K = 20$	$K = 30$	$K = 40$	$K = 50$
PCS	0.8839	0.8486	0.8370	0.8326	0.8418
ACS	0.7812	0.7585	0.7609	0.8029	0.9278
CS	0.7567	0.7601	0.7676	0.7751	0.7812
Proposed	0.7512	0.7437	0.7486	0.7557	0.7632

Table 8: MAE performance comparison of various item-based CF algorithms under different neighborhood size K with $N_c = 8$.

Algorithms	$N_c = 8$ ($P_{\text{pred}} = 67\%$)				
	$K = 10$	$K = 20$	$K = 30$	$K = 40$	$K = 50$
PCS	0.8839	0.8486	0.8370	0.8326	0.8418
ACS	0.7812	0.7585	0.7609	0.8029	0.9278
CS	0.7567	0.7601	0.7676	0.7751	0.7812
Proposed	0.7512	0.7437	0.7486	0.7557	0.7632

in the test set to evaluate the accuracy of the recommendation algorithms in terms of Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), which are computed as follows

$$\text{MAE} = \frac{1}{|\mathbb{T}|} \sum_{r_{ui} \in \mathbb{T}} |r_{ui} - \hat{r}_{ui}|,$$

$$\text{RMSE} = \sqrt{\frac{1}{|\mathbb{T}|} \sum_{r_{ui} \in \mathbb{T}} (r_{ui} - \hat{r}_{ui})^2},$$

where $\hat{r}_{ui}$ is the predicted rating, and r_{ui} is the actual rating in the test set denoted by $\mathbb{T}$. The smaller the MAE and RMSE, the better accuracy. Note that RMSE is more sensitive to large errors than MAE. In the experiments in Secs. 4.6.1 and 4.6.2, we assume all users are active and participate in BP during the same time slot. In Sec. 4.6.3 we examine the performance of the cascaded BP where only a subset of users are active during each time slot.

Table 9: RMSE performance comparison of various item-based CF algorithms under different neighborhood size K with $N_c = 3$.

Algorithms	$N_c = 3$ ($P_{\text{pred}} = 74\%$)				
	$K = 10$	$K = 20$	$K = 30$	$K = 40$	$K = 50$
PCS	1.0993	1.0562	1.0435	1.0392	1.0560
ACS	0.9896	0.9622	0.9671	1.0473	1.2712
CS	0.9754	0.9791	0.9876	0.9942	0.9992
Proposed	0.9680	0.9543	0.9580	0.9637	0.9703

Table 10: RMSE performance comparison of various item-based CF algorithms under different neighborhood size K with $N_c = 8$.

Algorithms	$N_c = 8$ ($P_{\text{pred}} = 67\%$)				
	$K = 10$	$K = 20$	$K = 30$	$K = 40$	$K = 50$
PCS	1.0993	1.0562	1.0435	1.0392	1.0560
ACS	0.9896	0.9622	0.9671	1.0473	1.2712
CS	0.9754	0.9791	0.9876	0.9942	0.9992
Proposed	0.9680	0.9543	0.9580	0.9637	0.9703

4.6.1 Performance Comparison

We compare our proposed privacy-preserving algorithm with other item-based CF algorithms using the well-known similarity measures, including the CS, PCS and ACS methods as introduced in Sec. 4.1.2. In all cases, rating predictions are generated by (32). We assume no privacy requirement is imposed on other algorithms, so the CS, PCS and ACS measures are directly applied to original rating data, and thus their results are not compromised by privacy techniques such as obfuscation. In particular, the presented results of the CS measure represent the best achievable performance of the distributed personal recommender system proposed in [76], as the item similarity between two items is computed based on the complete rating vectors associated with them in $\mathbb{R}$, rather than computed in an incremental manner as in [76].

The PCS and ACS methods compute the item similarity s_{ij} between two items i and j using the ratings from the set of their common users $U_{ij} = U_i \cap U_j$. We denote by N_c the minimum number of common users required for computing the

item similarity s_{ij} using PCS and ACS. If $|U_{ij}| < N_c$, then neither item i nor item j will be used to predict each other's ratings. Let $\mathbb{N}_i$ be the set of all valid items for item i under N_c. To predict ratings on item i for user u, the neighborhood $\mathcal{N}_{ui}$ used in (32) is formed from the set of items in $\mathbb{N}_{ui} = \mathbb{N}_i \cap I_u$. In addition, given a required neighborhood size K, if $|\mathbb{N}_{ui}| < K$, we simply say the unknown rating r_{ui} in the test set $\mathbb{T}$ is unpredictable by the PCS and ACS. We denote $\mathbb{T}_K$ as the subset of all predictable ratings in $\mathbb{T}$ at neighborhood size K. Note that $\mathbb{T}_K$ changes with N_c, since N_c impacts $\mathbb{N}_i$. For fair comparison of different algorithms, we apply the same N_c to all evaluated algorithms.

In Tables 7, 8 and 9, 10, we examine the MAE and RMSE performance of various algorithms. The parameters of the proposed algorithm are set as $D = 4$, $\mathcal{S} = \{1, 2\}$, and $\sigma = 0.5$. We show the results for $N_c = 3$ and 8 with K varying from 10 to 50 in steps of 10. Under each N_c, to fairly compare the performance of different K's, all results are obtained on the same subset of test ratings $\mathbb{T}_{50}$, and the percentage of ratings in $\mathbb{T}$ used for evaluation is

$$P_{\text{pred}} = |\mathbb{T}_{50}|/|\mathbb{T}| \times 100\%.$$

Our proposed algorithm achieves superior performance compared to other algorithms in terms of both MAE and RMSE. As K increases from 10, the performance of the proposed algorithm, as well as PCS and ACS algorithms, first improves but then degrades when K becomes too large, because ratings from neighbor items with smaller similarity to the active item, on which the rating is predicted, corrupt the prediction accuracy. Thus, to achieve the best performance, a proper neighborhood size K should be chosen. The computational complexity cost of our algorithm to solve for the similarities between item i and other items on graph $\mathcal{G}_i$ is $\mathcal{O}\left(NM^{2(1-\epsilon)}\right)$ for large N and M as discussed in Sec. 4.5, whereas for the CS method the complexity is $\mathcal{O}\left(NM\right)$, and for PCS and ACS, the complexity is $\mathcal{O}\left(NM^{(1-\epsilon)}\right)$.

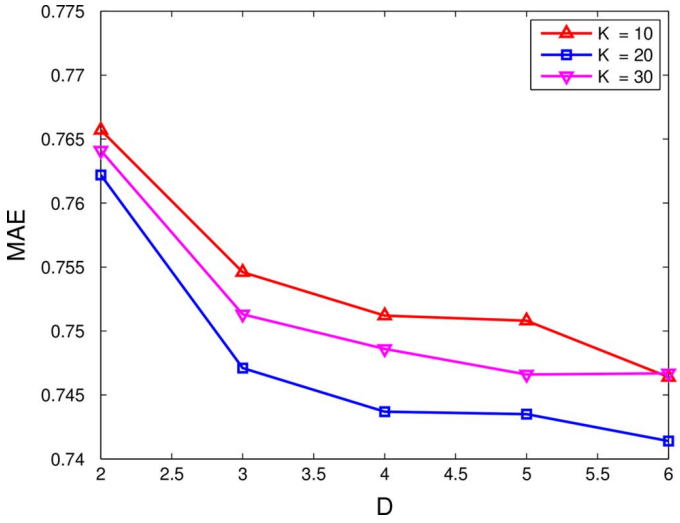


Figure 17: Impact of D on MAE of the proposed algorithm.

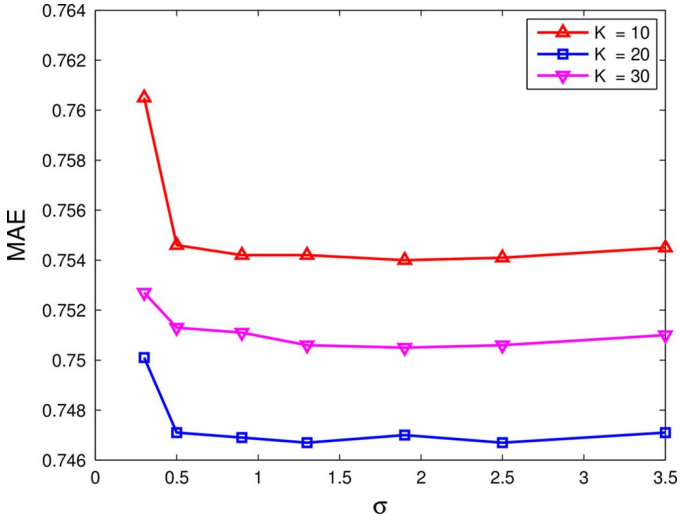


Figure 18: Impact of σ on MAE of the proposed algorithm.

4.6.2 Impact of Parameters

In the following experiments, we investigate the impact of the parameters D, σ, and $\mathcal{S}$ on the performance of our proposed algorithm. We always set $N_c = 3$ and evaluate the proposed algorithm on $\mathbb{T}_{50}$. In Fig. 17, we investigate the influence of group size D on the accuracy of the proposed algorithm, where we fix other parameters as $\mathcal{S} = \{1, 2\}$, and $\sigma = 0.5$. It can be seen that increasing D slightly improves accuracy. However, since the computational complexity at user side is $\mathcal{O}\left(\bar{N}DL^D\right)$, which is exponential

Table 11: Impact of $\mathcal{S}$ on MAE of the proposed algorithm.

$\mathcal{S}$	MAE		
	$K = 10$	$K = 20$	$K = 30$
$\mathcal{S}_2$	0.7546	0.7471	0.7513
$\mathcal{S}_5$	0.7504	0.7500	0.7545

in D, users need to wisely choose D according to their computational capability. In Fig. 18, we show the results for different σ's, where $D = 3$ and $\mathcal{S} = \{1, 2\}$. The performance slightly degrades if σ is too small or too large, since for a small σ, the curve of the factor function (37) quickly flattens out with respect to $|\hat{r}_{ui} - r_{ui}|$, while for a large σ, the curve becomes too flat. But overall, the algorithm is not sensitive for large dynamic ranges of σ.

In Table 11, we show the results of the proposed algorithm for different $\mathcal{S}$'s, where $\mathcal{S}_l = \{1, 2, \ldots, l\}, \forall l \in \{2, 5\}$, $D = 3$, and $\sigma = 0.5$. We observe that $\mathcal{S}_2$ and $\mathcal{S}_5$ achieve their best performance when $K = 20$, but $\mathcal{S}_2$ actually provides better accuracy than $\mathcal{S}_5$. This is because large l could cause overfitting, that is the obtained item similarity is strongly biased towards the memory data, and does not generalize well when used for prediction. Meanwhile, the computational complexity $\mathcal{O}\left(\bar{N}DL^D\right)$ at user side significantly increases with $L = |\mathcal{S}_l|$, depending on D.

4.6.3 Cascaded BP

We now investigate the more challenging scenarios where only a subset of users are available for BP at a given time slot. Firstly, we consider applying the basic BP algorithm without introducing priors, i.e., the inferred knowledge from previous time slots are not incorporated. The parameters are set as $\mathcal{S} = \{1, 2\}$, $D = 3$, $\sigma = 0.5$, and $K = \{10, 20, 30\}$. In Fig. 19, we show the MAE results with the percentage of participating users ranging from 10% to 100%. The performance degrades significantly as the percentage of users decreases. Thus, relying only on the users available at the current time slot is not satisfactory.

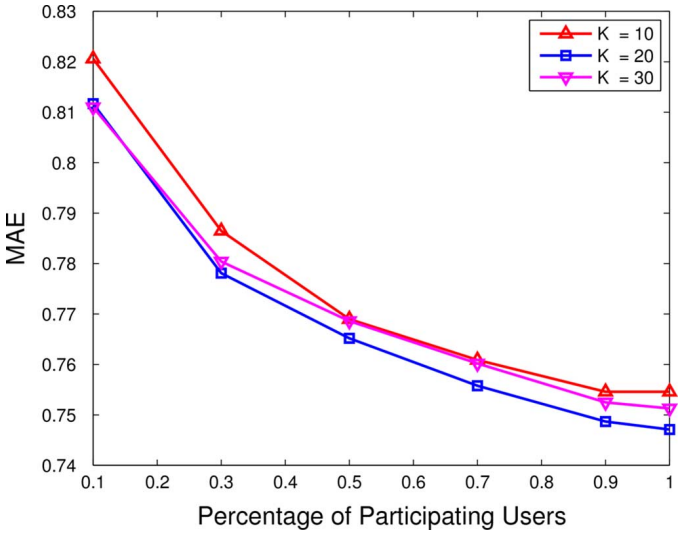


Figure 19: MAE versus percentage of participating users.

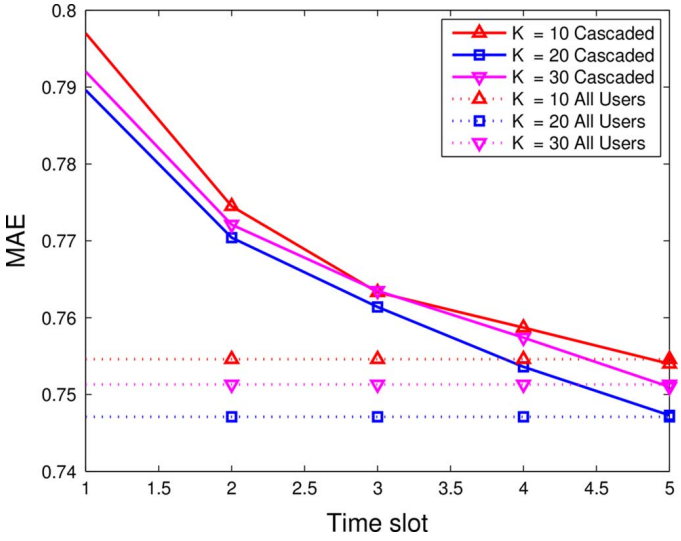


Figure 20: MAE performance of cascaded BP.

Next, we apply the cascaded BP proposed in Sec. 4.3.2. In the experiment, the users are randomly divided into 5 subsets with equal sizes, i.e., only 20% of users can participate in BP at each time slot. In Fig. 20, we show the MAE performance versus the number of elapsed time slots, where at each time slot the updated item similarity results are used to predict ratings. We also show the results of the original BP algorithm with all users participating at the same time slot. We can see that as the number of elapsed time slots increases, the MAE performance improves. After 5

time slots, the performance of the cascaded BP approaches that of the original BP algorithm with all users. Therefore, incorporating knowledge from previous time slots can effectively improve the performance.

4.7 Conclusion

In this chapter, we proposed a semi-distributed BP approach to item-based CF recommender system that preserves user privacy. The proposed algorithm computes item similarity by exchanging probabilistic messages between the server and users without directly exposing user ratings. To address the issue that only a subset of users participate in BP at one time slot, we proposed the cascaded BP scheme which accumulates the inferred knowledge from the previous time slots. With the computed similarities, a user then locally generates rating predictions as the weighted average of his own ratings on items. Through information-theoretic analysis, we showed that the proposed semi-distributed BP algorithm effectively preserves user privacy. In addition, we proposed a complexity reduction technique for efficient computation at the user side. The experimental results on the MovieLens dataset demonstrated that the BP algorithm with all users participating at the same time slot achieves superior performance, and the cascaded BP algorithm can improve performance as the number of elapsed time slots increases.

CHAPTER V

A FACTOR GRAPH APPROACH TO SHILLING ATTACK DETECTION

5.1 Introduction and Related Works

Recommender systems are increasingly employed by e-commerce websites, e.g., Amazon.com and Netflix.com, to provide personalized recommendations to users. Collaborative Filtering (CF) is so far the most popular recommendation algorithm, which relies on historic ratings given by users on items to make recommendations. Unfortunately, this approach is vulnerable to the so called "shilling" attacks [58], in which a group of spam users collaborate to manipulate the recommendations for their benefits, e.g., to recommend their products more often.

To protect the recommender systems against such attacks, one of the major approaches is to detect the spam users and remove them from the system. A number of detection algorithms have been proposed in the literature. Earlier work in [28] introduced several metrics for detecting the rating patterns of spam users. Then in [22], the authors improved the detection accuracy over [28] by including more model-specific features for classification. However, those feature-based algorithms suffer from low accuracy, as they only look at individual user rating patterns. In [73], the authors exploited the statistical properties of spam users, e.g., covariance, to perform detection via variable selection using Principle Component Analysis (PCA-VarSel). PCA-VarSel is very effective when spam users have low covariance, e.g., when they rate items randomly selected from all items, because genuine users have high covariance since they mostly rate only the popular items. However, it was shown in [47] that PCA-VarSel easily fails if spam users also selectively rate only those popular

items.

All the existing works focused extensively on rating patterns of users and detected spammers individually. In this work, we develop a probabilistic factor graph model that exploits the collaborative spamming behaviors among spammers to jointly detect them [129]. We use the Belief Propagation (BP) algorithm to perform inference efficiently.

5.2 Shilling Attack Detection

5.2.1 Attack Models

The shilling attack includes two general classes: push attacks and nuke attacks. In push attacks, the spam users collaboratively promote the target items by rating them high, whereas in nuke attacks, the spam users demote the target items by rating them low. In addition, to effectively affect the recommendations made to the genuine users, each spam user also rates a set of filler items to increase similarity with those genuine users. Some well-known shilling attack models include Random attacks and Average attacks [58]. In Random attacks, the set of filler items are rated randomly on a distribution learnt from the ratings in the dataset, whereas in Average attacks, each filler item is rated around the average rating of this individual item, making the spam users more similar to the genuine users. Hence, the Average attack is more effective than the Random attack. Moreover, in both attack models, the set of target items are given the highest allowed rating for push attacks and the lowest rating for nuke attacks.

5.2.2 Probabilistic Modeling of Attack Detection

We assume there are a set of M users denoted by $\mathbb{U} = \{1, \ldots, M\}$ and a set of N items denoted by $\mathbb{I} = \{1, \ldots, N\}$ in the system. Let r_{ui} denote user u's rating on item i, and $\mathcal{R}$ denote the set of all observed ratings in the system. The set of users who have rated item i is denoted by U_i, and the set of items rated by user u is denoted by

I_u. For each user u, we assign a binary variable m_u, where $m_u = 1$ if user u is a spam user and $m_u = 0$ otherwise. Similarly, for each item i, we assign a binary variable t_i, where $t_i = 1$ if item i is a target and $t_i = 0$ otherwise. Let $\mathcal{M} = \{m_u : 1 \leq u \leq M\}$ and $\mathcal{T} = \{t_i : 1 \leq i \leq N\}$. We denote by $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$ the joint posterior probability distribution of $\mathcal{M}$ and $\mathcal{T}$ given the observed ratings in $\mathcal{R}$. To identify spam users, we need to find the marginal distributions of $P(m_u|\mathcal{R})$, $\forall u \in \mathbb{U}$. $P(m_u|\mathcal{R})$ can be directly computed from $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$ by summing over all variables in $\mathcal{M}$ and $\mathcal{T}$ except m_u as follows

$$P(m_u|\mathcal{R}) = \sum_{\mathcal{M}\setminus m_u, \mathcal{T}} P(\mathcal{M}, \mathcal{T}|\mathcal{R}). \tag{69}$$

However, the computational complexity is $\mathcal{O}\left(2^{M+N}\right)$, which is exponential in the total number of users and items. In the following, we propose a proper factorization of $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$, and employ the efficient Belief Propagation (BP) algorithm that operates in a factor graph to infer $P(m_u|\mathcal{R})$, $\forall u \in \mathbb{U}$.

We begin by describing the local functions that $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$ factorizes into. Since spam users collaboratively work together to influence the ratings on target items, the local dependency among spam users is induced by their ratings on target items. Let $\mathcal{M}_i = \{m_u : u \in U_i, r_{ui} = r_a\}$, where r_a denotes the maximum rating in cases of push attacks and the minimum rating in cases of nuke attacks. The reader may focus on push attacks, as the method and results hold for the nuke attacks as well. Hence, we introduce the local function $f_i\left(\mathcal{M}_i, t_i|\mathcal{R}\right)$ to model the local probabilistic dependency among variables in $\{\mathcal{M}_i, t_i\}$. Given a configuration of $\mathcal{M}_i$, we can compute the rating bias caused by spam users as

$$\Delta r_i\left(\mathcal{M}_i\right) = \left| \frac{\sum_{u \in U_i} r_{ui}}{|U_i|} - \frac{\sum_{u \in U_i} r_{ui} - r_a \sum_{m \in \mathcal{M}_i} m}{|U_i| - \sum_{m \in \mathcal{M}_i} m} \right|, \tag{70}$$

where the first term in the absolute operator is the average rating on item i, including all ratings, and the second term is the unbiased average rating after removing ratings

from spam users. Based on (70), we express $f_i\left(\mathcal{M}_i, t_i|\mathcal{R}\right)$ as

$$f_i\left(\mathcal{M}_i, t_i|\mathcal{R}\right) = \frac{1}{1+\exp\left((-1)^{1-t_i}\alpha_t\left(\Delta r_i\left(\mathcal{M}_i\right)-\delta_r\right)\right)}, \tag{71}$$

where $\alpha_t < 0$ and $\delta_r > 0$, and both are adjustable scalars. Given a configuration of $\mathcal{M}_i$, if the rating bias $\Delta r_i\left(\mathcal{M}_i\right)$ exceeds δ_r, this function assigns a larger value for $t_i = 1$ than for $t_i = 0$, and vice versa.

We also introduce a local factor function for each variable in $\mathcal{M}$ and $\mathcal{T}$, so as to incorporate the local information extracted from each individual user and item. Let $g_u(m_u|\mathcal{R})$ be the local function for $m_u \in \mathcal{M}$. $g_u(m_u|\mathcal{R})$ can be designed to take into account the rating patterns of individual users, such as those user features introduced in [22]. Here, for the purpose of illustration, we consider the use of a single feature ϕ_u of user u. A simple probabilistic discriminative classifier [18] based solely on feature ϕ_u can be given in the following form

$$g_u(m_u|\mathcal{R}) = \frac{1}{1+\exp\left((-1)^{1-m_u}\beta_1(\phi_u-\tau_1)\right)}, \tag{72}$$

where β_1 and τ_1 are design parameters.

Likewise, we define a local factor function $h_i(t_i|\mathcal{R})$ for each $t_i \in \mathcal{T}$. We notice that the ratings of target items are more likely to have large variances than those of normal items, due to the eccentric ratings from spam users. Let φ_i denote the variance in ratings of item i. Similar to $g_u(m_u|\mathcal{R})$, we define $h_i(t_i|\mathcal{R})$ as follows

$$h_i(t_i|\mathcal{R}) = \frac{1}{1+\exp\left((-1)^{1-t_i}\beta_2(\varphi_i-\tau_2)\right)}, \tag{73}$$

where β_2 and τ_2 are design parameters.

Based on the local functions, we can express the factorization of $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$ as

$$P(\mathcal{M}, \mathcal{T}|\mathcal{R}) = \frac{1}{Z}\prod_{i\in\mathbb{I}} f_i\left(\mathcal{M}_i, t_i|\mathcal{R}\right)\prod_{u\in\mathbb{U}} g_u(m_u|\mathcal{R})\prod_{i\in\mathbb{I}} h_i(t_i|\mathcal{R}), \tag{74}$$

where Z is a normalization factor.

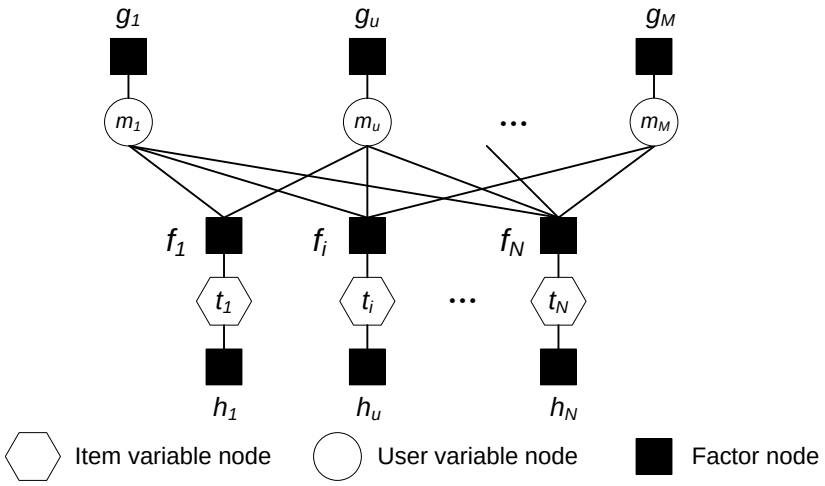


Figure 21: The factor graph for attack detection.

5.2.3 BP-based Inference in A Factor Graph

A factor graph is a bipartite graph that expresses the factorization structure of a function, where variable nodes and factor nodes represent variables and local functions, respectively, and an edge connects a variable node to a factor node if and only if the variable is an argument of the local function associated with the factor node [54]. We illustrate the factor graph that expresses the factorization of $P(\mathcal{M}, \mathcal{T}|\mathcal{R})$ by (74) in Fig. 21. Specifically, user variable m_u is represented by node m_u, and item variable t_i is represented by node t_i in Fig. 21. Each factor function in (74) is represented by a factor node. We connect each factor node g_i to item variable node t_i, and connect each factor node h_u to user variable node m_u. Also, we connect each factor node f_i to the user variable nodes in $\mathcal{M}_i$ and the item variable node t_i.

To infer the marginal distributions $P(m_u|\mathcal{R})$, $\forall u \in \mathbb{U}$, we apply BP in the factor graph to exploit the factorization for efficient inference. Since the constructed factor graph has loops, we cannot apply the standard BP algorithm for exact inference. We thus resort to loopy BP, which performs iterative message passing between variable nodes and factor nodes along the edges in the factor graph [54].

5.2.4 Complexity Reduction

While the BP algorithm can be much more efficient than direct computation using (69), the computational complexity for generating messages at the factor node is exponential in the degree of the factor node. This computational burden can be quite significant at factor node f_i, where the complexity in terms of multiplications is $\mathcal{O}\left(|\mathcal{M}_i|2^{|\mathcal{M}_i|}\right)$ for generating a message. Since in recommender systems an item can be rated by over hundreds of users, it renders the BP algorithm almost practically infeasible. Hence, we propose a complexity reduction technique by reducing the degree of factor node f_i as follows.

We randomly divide the user variable nodes in $\mathcal{M}_i$ into $G_i = \lceil |\mathcal{M}_i|/D \rceil$ groups, where D is a small integer. Let $\mathcal{M}_i^{(k)}$ denote the set of user variable nodes in group k, and M_{ik} be the size of group k, where $M_{ik} = D$ for $1 \leq k < G_i$, and $M_{ik} = |\mathcal{M}_i| \bmod D$ for $k = G_i$. Assuming independence among groups, we can approximate (74) using

$$P(\mathcal{M}, \mathcal{T}|\mathcal{R}) = \frac{1}{Z} \prod_{i \in \mathbb{I}} \prod_{k=1}^{G_i} f_i^{(k)} \left(\mathcal{M}_i^{(k)}, t_i|\mathcal{R}\right) \times \prod_{u \in \mathbb{U}} g_u(m_u|\mathcal{R}) \prod_{i \in \mathbb{I}} h_i(t_i|\mathcal{R}), \tag{75}$$

where we derive $f_i^{(k)} \left(\mathcal{M}_i^{(k)}, t_i|\mathcal{R}\right)$ from (71) by replacing $\Delta r_i\left(\mathcal{M}_i\right)$ with

$$\Delta r_i\left(\mathcal{M}_i^{(k)}\right) = \left| \frac{\hat{R}_i w_{ik} + r_a M_{ik}}{|\hat{U}_i| w_{ik} + M_{ik}} - \frac{\hat{R}_i w_{ik} + r_a \sum_{m \in \mathcal{M}_i^{(k)}} m}{|\hat{U}_i| w_{ik} + \sum_{m \in \mathcal{M}_i^{(k)}} m} \right|,$$

where $\hat{U}_i = \{u : u \in U_i, r_{ui} \neq r_a\}$, $\hat{R}_i = \sum_{u \in \hat{U}_i} r_{ui}$, and $w_{ik} = M_{ik}/|\mathcal{M}_i|$. Note that we allocate $\hat{R}_i$, the sum of ratings from $\hat{U}_i$, to each group in proportion to the group size. Similarly as in Sec. 5.2.3, we construct a new factor graph for (75) and apply the BP algorithm. The computational complexity at factor node $f_i^{(k)}$ is $\mathcal{O}\left(D2^D\right)$. The overall complexity of the BP algorithm with complexity reduction is $\mathcal{O}\left(D2^D|\mathcal{R}_a|\right)$, where $\mathcal{R}_a$ is the subset of the observed ratings with value r_a.

5.3 Experimental Evaluation

5.3.1 Experiment Setup

We evaluate the performance of the proposed BP-based attack detection algorithm using the 100K MovieLens dataset[1]. The dataset contains 100, 000 ratings, all integers from 1 to 5, on 1682 items (movies) by 943 users. We treat the original users in the dataset as genuine users. To launch shilling attacks, a number of spam users are injected into the system. The ratio of spam users to genuine users is set as $\frac{1}{10}$. Each spam user randomly selects a set of filler items from the top 50% most popular items according to the number of ratings each item receives, similar to [47]. We refer to the ratio of filler items to all items as *filler size*. We consider the Average attack model described in Sec. 5.2.1, since it is more effective than the Random attack. The rating on each filler item follows a normal distribution with standard deviation σ, and its mean is set as the average rating received by the filler item. Finally, a set of items are selected as targets in the attack. For the push attack, each target item has an average rating between 1 and 3, whereas for the nuke attack, each target item has an average rating between 3 and 5. We assume all spam users have the same set of targets.

We evaluate the performance of the attack detection algorithms in terms of *Precision* and *Recall* metrics, which are computed as follows

$$Precision = \frac{|\mathcal{D} \cap \mathcal{S}|}{|\mathcal{D}|} \quad \text{and} \quad Recall = \frac{|\mathcal{D} \cap \mathcal{S}|}{|\mathcal{S}|},$$

where $\mathcal{D}$ and $\mathcal{S}$ denote the sets of detected spam users and true spam users, respectively. We compare the performance of various detection algorithms, including the proposed BP algorithm, the PCA-VarSel algorithm in [73], and the feature-based algorithm using (72).

To assess the detection performance, the PCA-VarSel algorithm described in [73]

[1]Available at: http://www.grouplens.org/node/73.

sorts the users in ascending order of their contribution to the principle components (PCs) obtained from PCA analysis, and selects the top-M_d listed users as spam users. For easy comparison and fairness, in the proposed BP algorithm, we likewise sort users in descending order of $P(m_u = 1|\mathcal{R})$, and also select the top-M_d users as spam users. Similarly, in the feature-based algorithm, we select the top-M_d users ranked in descending order of $g_u(m_u = 1|\mathcal{R})$. In our experiments, we set M_d as the number of true spam users injected into the system, so the *Precision* and *Recall* are equal.

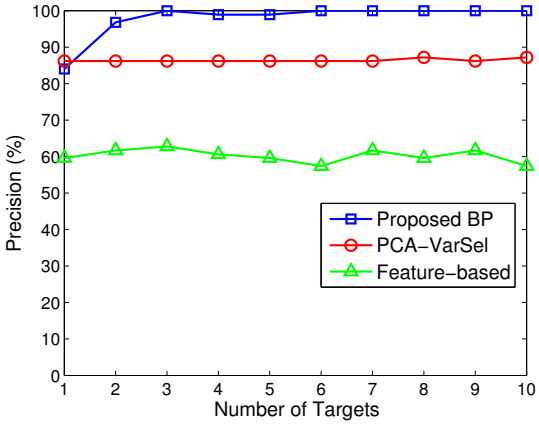
In the proposed BP algorithm, we adopt the MeanVar feature introduced in [22] for ϕ_u in (72). The MeanVar ϕ_u of user u is computed as

$$\phi_u = \frac{\sum_{i \in I_u \backslash \bar{I}_u} (r_{ui} - \bar{r}_i)^2}{|I_u \backslash \bar{I}_u|},$$

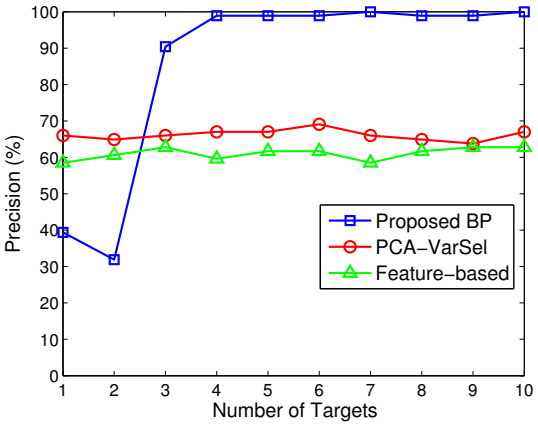
where $\bar{r}_i$ is the average rating of item i, and $\bar{I}_u = \{i : r_{ui} = r_a, i \in I_u\}$. The MeanVar feature is particularly effective for detecting spam users in Average attacks.

5.3.2 Results and Discussion

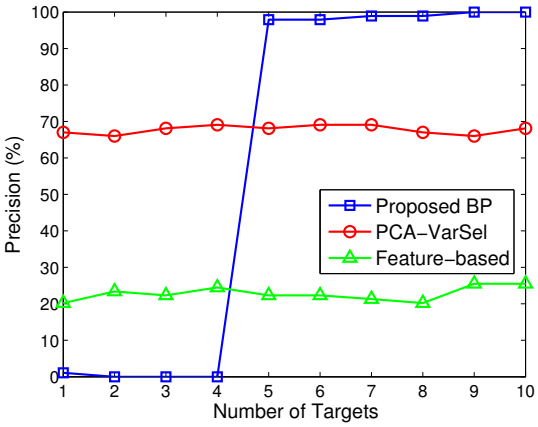
In Fig. 22, we present the results for detecting spam users in Push attacks, where we have set the parameters of the proposed BP algorithm as $\alpha_t = -3$ and $\delta_r = 0.35$ in (71), $\beta_1 = -1$ and $\tau_1 = 0.5$ in (72), $\beta_2 = 1$ and $\tau_2 = 1.5$ in (73), and the group size $D = 8$ for complexity reduction described in Sec. 5.2.4. The results show that the performance of the proposed BP algorithm improves significantly as the number of targets increases, reaching almost 100% precision when there are enough number of targets, whereas the other algorithms do not exhibit such improvement. This verifies that the proposed algorithm can effectively exploit the target items to detect spam users with high accuracy. Since the BP algorithm also incorporates the output $g_u(m_u|\mathcal{R})$ of the feature-based algorithm as illustrated in Fig. 21, we would like to examine the case when the performance of the feature-based algorithm is very poor. In Fig. 22c, the detection precision of the feature-based algorithm is only around 20% after we set $\sigma = 0.7$. Interestingly, we observe a threshold phenomenon for



(a) Filler size = 5% and $\sigma = 0.6$



(b) Filler size = 10% and $\sigma = 0.6$



(c) Filler size = 10% and $\sigma = 0.7$

Figure 22: Detection precision versus number of targets in *Push* attacks.

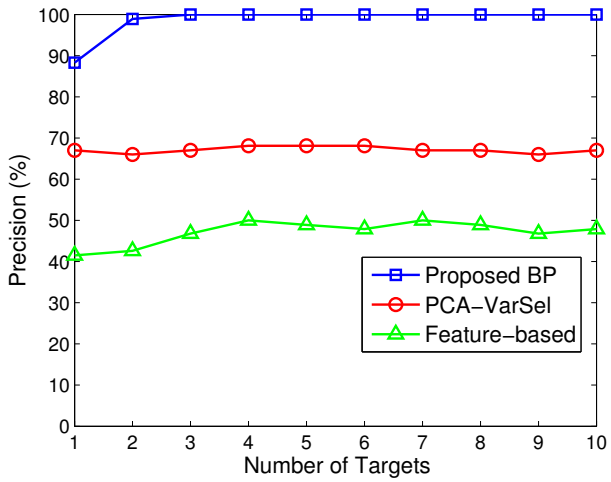
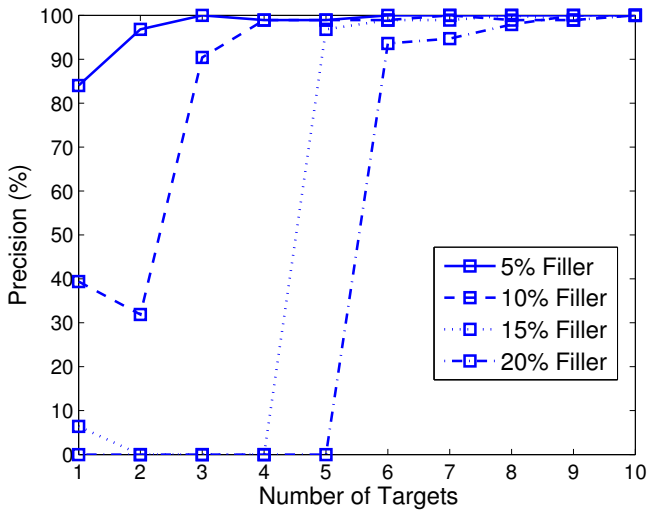


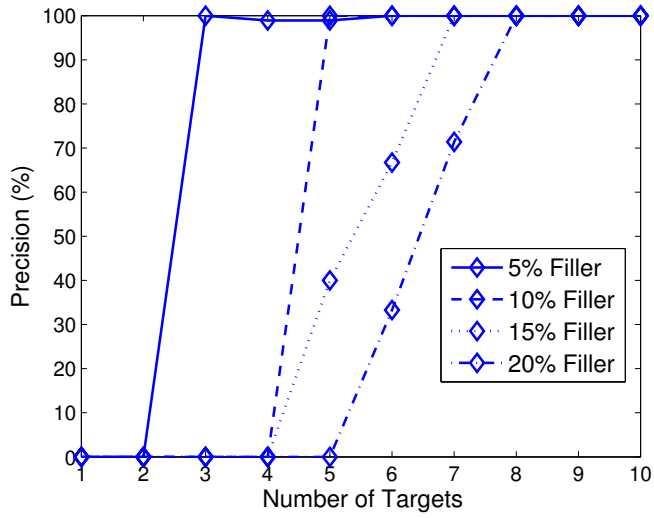
Figure 23: Detection precision versus number of targets in *Nuke* attacks. Filler size = 10% and $\sigma = 0.7$.

the BP algorithm, that is the BP algorithm has 0% precision when the number of targets is small, but when the number of targets exceeds a certain number, the BP algorithm provides almost 100% precision. This is because as the number of targets increases, the information gained from target items becomes dominant and corrects the results of the feature-based algorithm. Note that in practical implementations, we can incorporate into the BP algorithm more advanced feature-based algorithms for better performance. We have also performed experiments for Nuke attacks, and similar results can be observed, so we only present one set of those results in Fig. 23.

In Fig. 24, we investigate the impact of the filler size on the performance of the BP algorithm. We show the detection results for both spam users and target items in Fig. 24a and Fig. 24b, respectively. Here, to evaluate the detection performance for the target items, we infer $P(t_i|\mathcal{R})$ from (74) and select the top-N_t items ranked in descending order of $P(t_i = 1|\mathcal{R})$, where N_t is set as the number of true targets. Note that the existing algorithms only detect spam users. As the filler size increases, the BP algorithm suffers from performance loss for small numbers of targets. This is because the BP algorithm relies on information from target items, but the noise information from the filler items corrupts the useful information. Indeed, we can see



(a) Detection precision of spam users versus number of targets



(b) Detection precision of target items versus number of targets

Figure 24: Detection performance in *Push* attacks under varying filler sizes. $\sigma = 0.6$.

in Fig. 24b that the detection precision for target items drops with increasing filler size. However, when there are enough target items to suppress the noise information, the BP algorithm still can achieve high detection precision. It is worth noting that in practice a higher cost is incurred for attackers to increase the filler size while decreasing the target number. Also, the BP algorithm can be enhanced with feature-based algorithms that deliver high detection accuracy for large filler sizes.

5.4 Conclusion

To protect the collaborative filtering systems against the shilling attacks, we proposed a BP-based algorithm for jointly detecting spam users in a probabilistic inference framework. Different from the existing algorithms that rely solely on users' rating patterns, the proposed algorithm further exploits the colluding spammers' collaborative spamming behaviors towards target items. Considering the computational complexity for inference, we developed a factorized probabilistic model for attack detection and applied BP to perform inference efficiently. The proposed detection algorithm can also conveniently incorporate the existing algorithms for enhanced performance. We showed through experiments that the proposed BP algorithm significantly improves detection accuracy as the number of target items increases. We also observed that its performance degrades with increasing filler sizes. In the future work, we will develop more robust algorithms to deal with more sophisticated attack scenarios.

CHAPTER VI

EXPLOITING POPULARITY AND SIMILARITY FOR SOCIAL LINK RECOMMENDATION

6.1 Introduction

Twitter is a popular on-line platform for social networking and information sharing. Twitter users can follow other users to receive messages, or tweets, from them. However, given the limited attention and time, most users would like to follow only the most relevant users. Due to the huge number of users in Twitter, recommendation algorithms are needed to help users automatically discover new interesting users to follow.

Many existing link recommendation algorithms for social networks are developed with focus on the link structure [64]. The simplest example is to recommend the most popular users with the largest number of connections. Other common algorithms first weigh each link by some importance score, and rank the nodes according to the sum of importance scores of their links, e.g., the PageRank algorithm [84]. Twitter provides the user recommendation service called WTF ("Who To Follow") [41], which was built on SALSA (Stochastic Approach for Link-Structure Analysis) [61], a random-walk algorithm similar to PageRank. We generally refer to those algorithms as "popularity" or "weighted popularity" based algorithms.

Yet, unlike other social networks such as Facebook, besides to establish social connections, many Twitter users follow other users to receive information interesting to them. Hence, it is promising to exploit the similarity between Twitter users for recommendation, i.e., to recommend other users similar to the followees already followed by the follower, or to recommend other users who are similar to the follower.

In [43], the authors proposed a content-based algorithm that match user interests by directly analyzing the texts of user tweets. In [75], the authors proposed a collaborative filtering algorithm using the matrix factorization technique to learn latent user interests from the user feedbacks, e.g., to follow a user or not.

Indeed, a recent study has shown that popularity and similarity are the two important factors that drive the growth of a variety of networks including the Internet and social networks [85]. In this work, we compare various popularity-based algorithms and similarity-based algorithms for Twitter user recommendation, and propose two approaches to exploiting both popularity and similarity [131]. The first approach is to employ rank aggregation techniques [32]; the second approach is to adapt the collaborative filtering algorithms to incorporate popularity in addition to similarity.

6.2 Popularity versus Similarity

In this section, we describe various popularity-based algorithms and similarity-based algorithms.

6.2.1 Problem Description

To recommend new users for the follower to follow, the recommendation algorithm generates a personalized ranked list of users. However, due to the huge number of users in Twitter, it is too costly to compute a ranking of network-wide users for making personalized recommendations to each follower. Many algorithms instead recommend users from the local network that centers around the follower [41]. The empirical study in [124] revealed that 90% of new links in Twitter-like microblogging networks go to users two hops away from the follower, i.e., followees of the follower's followees.

As such, in the rest of this paper, we focus on recommending users from 2-hop users for the follower. We denote the follower whom to recommend users for as source s, the set of its initial followees as $\mathcal{F}_1$, and the set of all followees of $\mathcal{F}_1$ as $\mathcal{F}_2$. We

also denote $F^+(s)$ as the subset of $\mathcal{F}_2$ that source s follows as network grows. The recommendation problem is essentially to compute a ranked list of users from $\mathcal{F}_2$ for source s.

6.2.2 Recommendation Algorithms

6.2.2.1 Popularity-based algorithms

For the popularity based or weighted popularity based algorithms, we consider Adamic-Adar score [64] and SALSA [61]. Adamic-Adar score is a simple and effective measure for link recommendation. We adopt it for Twitter user recommendation, and compute the score for a 2-hop user $u \in \mathcal{F}_2$ of source s as

$$AA(s,u) = \sum_{z \in \mathcal{P}(u) \cap \mathcal{F}_1} \frac{1}{|\mathcal{R}(z)|}, \tag{76}$$

where $\mathcal{P}(u)$ denotes the set of followers of user u, and $\mathcal{R}(z)$ denotes the set of followees of user z.

In the SALSA algorithm, we construct a bipartite graph $\mathcal{G}$ by assigning $\mathcal{F}_1$ as "hubs" and $\mathcal{F}_2$ as "authorities". The algorithm performs two distinct forward-backward random walks starting from the "authorities" side or the "hubs" side. The scores for the authorities and hubs are related as

$$a_i = \sum_{\{k|(k,i)\in\mathcal{G}\}} \frac{h_k}{|\mathcal{R}(k)|}, \quad h_k = \sum_{\{i|(k,i)\in\mathcal{G}\}} \frac{a_i}{|\mathcal{P}(i)|} \tag{77}$$

where a_i and h_k denote the scores of authority node i and hub node k, respectively, and (k,i) denotes an edge in bipartite graph $\mathcal{G}$. The users in $\mathcal{F}_2$ ("authorities") are ranked by the authority scores.

6.2.2.2 Similarity-based algorithms

For the similarity-based algorithms, we focus on the collaborative filtering recommendation algorithms using the observed follower-followee relationship between users as implicit feedback, including the neighborhood method and the MF-BPR (Matrix

Factorization with Bayesian Personalized Ranking) method [94]. The basic principle is to recommend to source s the followees of the $\mathcal{F}_1$ users who share similar interests with source s. Using the terms of traditional user-item recommender systems, the set of "users" corresponds to $\mathcal{U} = \{s\} \cup \mathcal{F}_1$, and the set of "items" corresponds to $\mathcal{F}_2$. Moreover, the established follower-followee relationships between $\mathcal{U}$ and $\mathcal{F}_2$ are considered as observed implicit feedback on "items" in $\mathcal{F}_2$ from "users" in $\mathcal{U}$. The collaborative filtering approach exploits the collective implicit feedbacks to generate recommendations.

In the neighborhood method, we compute the similarity between source s and other users $u \in \mathcal{F}_1$ using Jaccard's coefficient as

$$J(s,u) = \frac{|F^+(u) \cap F^+(s)|}{|F^+(u) \cup F^+(s)|} \tag{78}$$

where $F^+(u)$ denotes the subset of $\mathcal{F}_2$ followed by user u. The score on user $i \in \mathcal{F}_2$ is predicted as

$$R(s,i) = \sum_{u \in \mathcal{F}_1} J(s,u) R(u,i), \tag{79}$$

where $R(u,i) = 1$ if u follows i and $R(u,i) = 0$ otherwise.

The MF-BPR method learns a model for correctly ranking pairs of users in $\mathcal{F}_2$. It is assumed that if user u follows user i but not user j, $\forall i,j \in \mathcal{F}_2$, then user u prefers user i over j. We represent such relationship as (u,i,j). The training data is created as $\mathcal{D} = \{(u,i,j) | i \in F^+(u), j \in \mathcal{F}_2 \backslash F^+(u)\}$. We associate each user u with a K-dimensional latent vector $\mathbf{w}_u$, and each $i \in \mathcal{F}_2$ with a K-dimensional latent vector $\mathbf{v}_i$. Let W and V represent the collections of all $\mathbf{w}_u$ and all $\mathbf{v}_i$, respectively. The probability that user u prefers i over j is defined as

$$P(i >_u j | W, V) = \frac{1}{1 + e^{-(X(u,i) - X(u,j))}}, \tag{80}$$

where $X(u,i) = \langle \mathbf{w}_u, \mathbf{v}_i \rangle$, and $\langle , \rangle$ indicates the dot product. We further introduce a normal distribution $\mathcal{N}(\mathbf{0}, \lambda^{-1} I_K)$ as the prior distribution for each $\mathbf{w}_u$ and $\mathbf{v}_i$, where

$\lambda > 0$ and I_K is a $K \times K$ identity matrix. The latent vectors are learnt by maximizing the posterior probability, which can be formulated as

$$\max_{W,V} \quad \log \prod_{(u,i,j)\in\mathcal{D}} P(i >_u j|W,V)P(W)P(V). \tag{81}$$

We apply the stochastic gradient-descent algorithm with bootstrap sampling as in [94] to solve this optimization problem. The latent vectors are updated for triple $(u,i,j) \in \mathcal{D}$ with learning rate μ as follows

$$\mathbf{w}_u \leftarrow \mathbf{w}_u + \mu \left(\frac{\mathbf{v}_i - \mathbf{v}_j}{1 + e^{X(u,i)-X(u,j)}} - 2\lambda \mathbf{w}_u \right), \tag{82}$$

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \mu \left(\frac{\mathbf{w}_u}{1 + e^{X(u,i)-X(u,j)}} - 2\lambda \mathbf{v}_i \right), \tag{83}$$

$$\mathbf{v}_j \leftarrow \mathbf{v}_j + \mu \left(\frac{-\mathbf{w}_u}{1 + e^{X(u,i)-X(u,j)}} - 2\lambda \mathbf{v}_j \right). \tag{84}$$

To recommend users for source s, we compute $X(s,i)$, $\forall i \in \mathcal{F}_2$, and rank the users in descending order of $X(s,i)$.

6.3 Link Recommendation Using Popularity and Similarity

We propose two approaches to exploiting both popularity and similarity: rank aggregation and popularity-biased collaborative filtering. The rank aggregation approach combines the recommendation results of the popularity-based algorithm and the similarity-based algorithm, while each individual algorithm generates recommendations independently. It is worth noting that the scores predicted by different recommendation algorithms can be very different in both the range and the form, which are not suitable for aggregation, and hence we only consider order-based rank aggregation that combines the rankings which can be easily derived by sorting users by the predicted scores.

The popularity-biased collaborative filtering approach directly adapts the original collaborative filtering algorithms to incorporate popularity in addition to similarity. We introduce two specific cases of the neighborhood algorithm and the MF-BPR algorithm.

6.3.1 Rank Aggregation

Rank aggregation combines several ranking lists to generate a "consensus" ranking. It has been widely applied in Web search and document retrieval, e.g., meta-search engines and multi-criteria search. There are various rank aggregation algorithms such as Borda's method, median rank aggregation, and Markov chain methods. We choose the Markov chain method for its superior performance. In particular, we focus on the MC3 method among the four Markov chain methods proposed in [32].

Suppose we have two ranked lists of users denoted by τ_1 and τ_2, which are generated using the popularity-based algorithm and the similarity-based algorithm, respectively. The Markov chain rank aggregation method assigns a unique state to each of the users, and specifies the transition matrix as follows: If the current state is user i, randomly pick a ranking list τ with probabilities $P(\tau = \tau_1) = \alpha$ and $P(\tau = \tau_2) = 1 - \alpha$, then uniformly pick a user j from τ, and go to j if $j >_\tau i$ else stay in i. Here, $j >_\tau i$ means the list τ ranks j closer to the top than i. Let T_k represent the transition matrix for the Markov chain derived from individual ranking list τ_k, $k = 1, 2$, where $T_k(i, j)$ is the transition probability from state i to j,

$$T_k(i,j) = \begin{cases} \frac{1}{|\mathcal{F}_2|}, & \text{if } j >_{\tau_k} i \\ 1 - \frac{|\{j | j >_{\tau_k} i\}|}{|\mathcal{F}_2|}, & \text{if } j = i \\ 0, & \text{otherwise} \end{cases} \tag{85}$$

The transition matrix T of the rank aggregation Markov chain can be written as

$$T = \alpha T_1 + (1 - \alpha) T_2. \tag{86}$$

The aggregated ranking is obtained by ranking users according to the stationary probabilities of the Markov chain, which can be computed by solving

$$T^\top \boldsymbol{\pi} = \boldsymbol{\pi}, \quad \text{s.t.} \quad \sum_{i \in \mathcal{F}_2} \pi_i = 1, \ \pi_i > 0 \tag{87}$$

where $\boldsymbol{\pi}$ denotes the stationary probability vector.

By varying the parameter α between 0 and 1, we can bias the aggregation rank towards similarity or popularity, with $\alpha = 0$ for similarity and $\alpha = 1$ for popularity. Further, α can be personalized for individual users to better predict their behavior and thus meet individual preferences.

6.3.2 Popularity-biased Collaborative Filtering

We adapt the two cases of the neighborhood method and the matrix factorization method, which are the most popular collaborative filtering recommendation algorithms, for popularity-biased collaborative filtering. It should be noted that the specific techniques proposed here might not be directly applicable to other collaborative filtering algorithms, as they may have very different prediction models. Nevertheless, we stress the viability and benefits of incorporating popularity into collaborative filtering link recommendation.

6.3.2.1 The popularity-biased neighborhood method

The neighborhood method predicts the score on user $i \in \mathcal{F}_2$ for source s using (79), where the feedback $R(u, i)$ from user u in $\mathcal{F}_1$ (set as 1 for following i and 0 for not following i) is weighted by the similarity of user u to source s. However, the similarity computed using the Jaccard's coefficient assigns zero to users who do not have common followees with the source. To bias the neighborhood method towards popularity, we modify the similarity measure as follows

$$J_p(s, u) = \frac{|F^+(u) \cap F^+(s)| + c}{|F^+(u) \cup F^+(s)|}, \tag{88}$$

where $c > 0$ ensures that every user $u \in \mathcal{F}_1$ is counted with $J_p(s, u) > 0$. In this way, the popular users who are followed by many users in $\mathcal{F}_1$, though not necessarily highly similar to the source, can have a higher predicted score than other users followed by only a few users with high similarity to the source.

6.3.2.2 The popularity-biased MF-BPR method

The MF-BPR method represents each user $i \in \mathcal{F}_2$ using a latent vector as described in Sec. 6.2.2.2. The similarity of user interests is modelled by the angle between latent vectors. We now adapt the model to also incorporate the popularity of each user by the length, or magnitude, of the latent vector. We assume a prior distribution $\mathcal{N}(\gamma\mathbf{1}, \lambda^{-1}I_K)$ for the latent vectors, where $\mathbf{1}$ denotes a vector with all entries one, and randomly initialize the latent vectors following the normal distribution $\mathcal{N}(\gamma\mathbf{1}, \beta I_K)$, where γ needs to be chosen relatively large. Using the stochastic gradient descent learning algorithm, the latent vectors are updated for triple $(u, i, j) \in \mathcal{D}$ as follows

$$\mathbf{w}_u \leftarrow \mathbf{w}_u + \mu \left(\frac{\mathbf{v}_i - \mathbf{v}_j}{1 + e^{X(u,i)-X(u,j)}} - 2\lambda(\mathbf{w}_u - \gamma\mathbf{1}) \right), \tag{89}$$

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \mu \left(\frac{\mathbf{w}_u}{1 + e^{X(u,i)-X(u,j)}} - 2\lambda(\mathbf{v}_i - \gamma\mathbf{1}) \right), \tag{90}$$

$$\mathbf{v}_j \leftarrow \mathbf{v}_j + \mu \left(\frac{-\mathbf{w}_u}{1 + e^{X(u,i)-X(u,j)}} - 2\lambda(\mathbf{v}_j - \gamma\mathbf{1}) \right). \tag{91}$$

The latent vector $\mathbf{v}_i$ of user i is updated through (90), where the angle between $\mathbf{v}_i$ and $\mathbf{w}_u$ is expected to be small, as they are both close to the vector $\gamma\mathbf{1}$ if γ is relatively large. Hence, $\mathbf{v}_i$ increases in length after each update. As popular users are updated more often in the stochastic optimization process, their associated latent vectors will have larger length.

The score on user $i \in \mathcal{F}_2$ by source s is predicted by

$$X(s, i) = \langle \mathbf{w}_s, \mathbf{v}_i \rangle = \cos\theta_{si} |\mathbf{w}_s| |\mathbf{v}_i|, \tag{92}$$

where θ_{si} is the angle between $\mathbf{w}_s$ and $\mathbf{v}_i$, which reflects the similarity of interests between user i and source s. Moreover, with the popularity-biased MF-BPR method, the magnitude $|\mathbf{v}_i|$ is large for popular users. Therefore, both similarity and popularity are taken into account. Also, increasing γ biases the algorithm more towards popularity.

6.4 Experimental Evaluation

In this section, we compare the empirical performance of all recommendation algorithms on the Twitter dataset and the Tencent Weibo dataset. Tencent Weibo is a Twitter-like online microblogging service widely used in China. The Twitter dataset is downloaded from Twitter.com using the Twitter API. The Tencent Weibo dataset is a subset of the dataset provided by KDD Cup 2012 Track 1[1]. The source users are selected such that they have at least 100 followees. For each source user, we use its earliest 50 followees as the initial set $\mathcal{F}_1$, and construct the set $\mathcal{F}_2$ from the followees of $\mathcal{F}_1$. The statistics of the two datasets are shown in Table 12, where the number of $\mathcal{F}_2$ users is the total of $\mathcal{F}_2$ users from all source users, and the number of edges refers to the total number of follower-followee relationships between users.

In the experiment, we use 70% of the subset of $\mathcal{F}_2$ followed by each source user for training, 15% for validation, and 15% for testing. To reduce computational complexity, we further prune the $\mathcal{F}_2$ users that are followed by less than 5 users, as they are very unlikely to be followed by the source users. The metrics used for evaluating recommendation performance are AUC (Area Under the ROC Curve) and P@10 (Precision at top 10). The AUC metric measures the algorithm's overall ability to rank positive instances above negative instances, whereas the P@10 metric emphasizes more on the quality of the top 10 recommended instances. We report results of all algorithms on the testing set.

[1] `http://www.kddcup2012.org/c/kddcup2012-track1`

Table 12: Statistics of the two datasets.

Dataset	# of sources	# of $\mathcal{F}_2$ users	# of edges
Twitter	158	3,195,481	6,822,720
Tencent	200	22,904	195,760

Table 13: Popularity versus similarity.

Algorithms	Twitter dataset		Tencent dataset	
	AUC	P@10	AUC	P@10
Adamic-Adar	0.7822	0.0144	0.7351	0.0934
SALSA	0.7458	0.0258	0.7168	0.1757
Neighborhood	0.6498	0.0186	0.6253	0.1326
MF-BPR	0.6363	0.0196	0.6730	0.1459

6.4.1 Popularity versus Similarity

We first compare the performance of the popularity-based algorithms (Adamic-Adar and SALSA) and the similarity-based algorithms (Neighborhood and MF-BPR) as shown in Table 13. The parameters of the MF-BPR algorithm are set as $K = 20$, $\lambda = 0.01$ and $\mu = 0.001$. The results show that the popularity-based algorithms generally achieve better AUC than the similarity-based algorithms. However, the similarity-based algorithms have quite good P@10 performance. Overall, combining the results suggests that while Twitter users tend to follow popular users, they also like to follow users with similar interests to them. Hence, there is a trade-off between popularity and similarity.

6.4.2 Combining Popularity and Similarity

For the rank aggregation approach, we experiment with different combinations of popularity-based and similarity-based algorithms, including AA & Neighb (Adamic-Adar & Neighborhood), AA & MF-BPR (Adamic-Adar & MF-BPR), SLS & Neighb (SALSA & Neighborhood), and SLS & MF-BPR (SALSA & MF-BPR). The parameter α is selected independently for each individual user such that the AUC is maximized on the validation set for that user.

For the popularity-biased collaborative filtering approach, we evaluate the Pop-Neighb (Popularity-biased Neighborhood) algorithm and the Pop-MF-BPR (Popularity-biased MF-BPR) algorithm. The parameter c is set to 1 for Pop-Neighb, and the

Table 14: Performance of algorithms combining popularity and similarity.

Algorithms	Twitter dataset		Tencent dataset	
	AUC	P@10	AUC	P@10
AA & Neighb	0.7873	0.0155	0.7458	0.1365
AA & MF-BPR	0.7835	0.0227	0.7540	0.1519
SLS & Neighb	0.7493	0.0247	0.7156	0.1746
SLS & MF-BPR	0.7606	0.0309	0.7392	**0.1862**
Pop-Neighb	0.7746	0.0186	0.7236	0.1657
Pop-MF-BPR	**0.7940**	**0.0536**	**0.7734**	0.1818

parameters for Pop-MF-BPR are set as $\gamma = 1$ and $\beta = 0.01$, while other parameters are set the same as in the original MF-BPR algorithm.

The results in Table 14 show that the popularity-biased MF-BPR algorithm achieves the best performance in terms of both AUC and P@10. Comparing the results of rank aggregation algorithms with those of the individual algorithms in Table 13, we can see that rank aggregation can combine the advantages of the popularity-based algorithm and the similarity-based algorithm. The popularity-biased neighborhood method also improves over the original neighborhood method. In summary, the evaluation results confirm that combining popularity and similarity can improve the overall recommendation performance in terms of AUC as well as the quality of the top ranked recommendations.

6.5 Conclusion

We proposed two approaches, the rank aggregation approach and the popularity-biased collaborative filtering approach, to exploiting both popularity and similarity for Twitter user recommendation. The rank aggregation approach combines the ranked recommendations generated by the popularity-based algorithm and the similarity-based algorithm. The popularity-biased collaborative filtering approach adapts the Bayesian personalized ranking method to incorporate popularity. Through experimental evaluation on two real-world datasets, we showed that the rank aggregation

algorithms can combine the advantages of popularity-based and similarity-based algorithms, and the popularity-biased collaborative filtering algorithms improve upon the original algorithms in terms of both AUC and P@10.

CHAPTER VII

REAL-TIME SOCIAL MEDIA EVENT CREDIBILITY PREDICTION

7.1 Introduction and Related Works

Online social media services like Twitter are widely adopted by people to self-report activities and stories happening around them [56]. Monitoring social media streams, e.g., tweets in Twitter, becomes an effective way to detect real-time events and monitor emergent situations [98, 112]. However, social media is also increasingly exploited to spread rumors and false information, e.g., fake images during Hurricane Sandy [41].

Automatic methods for assessing event credibility in social media services were studied in [26], wherein the authors explored various aggregation features given a set of tweets related to an event, and employed machine learning tools, e.g., Decision Tree, for predicting event credibility. In [57], the authors also identified other prominent features including temporal, structural, and linguistic features. However, those features proposed in the above works require almost a complete set of tweets related to the event, which usually means continuously collecting tweets over a long period of time, causing significant delays in the prediction task from the time the event first takes place.

The work in [40] proposed methods for real-time credibility assessment of individual tweets, where a semi-supervised ranking model was used to assign credibility scores to tweets in a user's timeline based on features derived from single tweets. Since it only requires the data of each individual tweet without assuming the complete data of an event, it is able generate results in real time. Another work in [39] studied the

credibility of tweets during high impact events. But neither of the works addressed the real-time prediction of event credibility.

The interaction between the social media community and online information has been studied in many works. Various algorithms were proposed to predict whether a tweet will be retweeted [90, 107, 121], whether a tweet will be retweeted by a given user [115, 16], the number of retweets of a tweet [45, 55], and who will retweet a message in Twitter [66]. The work in [79] investigated the mechanism of credibility perception by users. It revealed that users are not good at judging information credibility based only on the message content, rather their perceptions are also influenced by author credentials. However, those works did not consider utilizing the community interactions towards event credibility analysis.

Streaming processing of tweets was studied in [89, 31, 109]. In [89], the authors proposed a streaming model of computation for detecting new events from a stream of Twitter posts. They achieved significant speedup in processing the large volume of data coming from Twitter. The work in [31] predicted latent user attributes in Twitter with stream-based classification, and it showed that the bag-of-words classification model can be decomposed into a series of streaming updates. The work in [109] applied Bayesian rule update to dynamical models for real-time streaming prediction of political preferences of Twitter users.

In this chapter, we develop a probabilistic generative model for real-time event credibility prediction with streaming tweets [134]. The model depicts the generative process of individual tweets. Tweets related to true and false events are drawn from different distributions. In addition, the model also captures the interaction between the social media community and tweets. The research in [74] on information propagation in Twitter showed that tweets related to false rumors are propagated differently from tweets of true news because rumors are more likely to be questioned and denied by the Twitter community.

Based on the generative model, we propose an online algorithm that updates prediction with streaming tweets. In contrast to offline aggregation analysis that requires a complete set of tweets related to an event, the proposed algorithm only uses the currently observed streaming tweets. The algorithm incrementally updates prediction without the need to store or reprocess the past tweets. We conduct experiments on the dataset of tweets collected from Twitter. The results show that the online prediction performance quickly approaches that of the batch prediction with only a few hundred tweets.

7.2 Generative Model

In Twitter, users post messages, also known as tweets, to describe happening events around them. Previous research in [26] has shown that the tweets related to true events and false events exhibit different characteristics, which can be exploited for automatic event credibility analysis. Meanwhile, the social media community interact with messages by retweeting and favoriting messages. According to the study in [74], the community interact differently with tweets related to true and false events, as messages related to false events are questioned and denied more than messages related to true events.

Our goal is to predict the credibility label of a new event with the relevant messages collected from the streaming tweets in Twitter. In particular, we hope to predict the label in real time, that is, to start prediction after observing only a few tweets in the early stage, rather than wait until collecting a complete set of tweets over a long period of time. In the following, we introduce a generative probabilistic model that describes the generative process of tweets related to true and false events as well as the community interactions, and specify the dependency relationships between various variables. Using the generative model, we develop a real-time online prediction algorithm with streaming tweets.

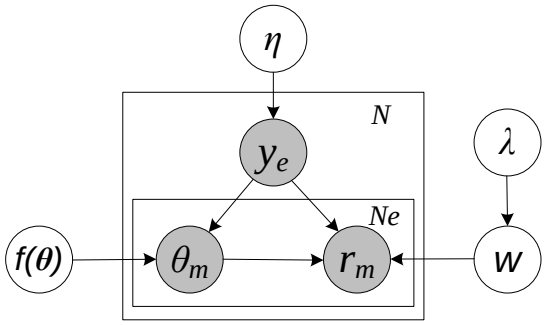


Figure 25: Graphical representation of the generative model for event credibility prediction.

7.2.1 Generative Process

We denote an event in Twitter as e, and use y_e to denote its label, $y_e = 1$ if true and $y_e = 0$ if false. The label y_e is modelled by $y_e \sim \text{Bernoulli}(\eta)$, where η can be used to specify prior knowledge about labels. In this work, we always set $\eta = 0.5$, i.e., no prior information is known. We define a set of messages related to event e as $M_e = \{m_1(e), m_2(e), \ldots, m_{N_e}(e)\}$, where N_e is the number of observed messages. Each message m can be characterized by a set of features represented as $\boldsymbol{\theta}_m = [\theta_1, \ldots, \theta_L]$, where L is the number of features. The features can be directly extracted from the individual messages. To model feature vector $\boldsymbol{\theta}_m$ related to true and false events, we use a mixture distribution $f(\boldsymbol{\theta})$, where

$$f(\boldsymbol{\theta}|y_e) = y_e f_1(\boldsymbol{\theta}) + (1 - y_e) f_0(\boldsymbol{\theta}). \tag{93}$$

Given the event is true, $y_e = 1$, $\boldsymbol{\theta}_m$ is drawn from the distribution $f_1(\boldsymbol{\theta})$, otherwise $\boldsymbol{\theta}_m$ is drawn from the distribution $f_0(\boldsymbol{\theta})$. We use non-parametric density estimation methods to learn $f(\boldsymbol{\theta})$ from training data.

The community interaction on message m is denoted by r_m, $r_m = 1$ for positive feedbacks and $r_m = 0$ for negative feedbacks. We consider a message receives positive feedback if the total number of "Retweets" and "Favorites" is greater than a preset threshold, and negative feedback vice versa. The community feedback r_m is modelled as $r_m \sim \text{Bernoulli}(p_m)$, where p_m depends on both the massage feature $\boldsymbol{\theta}_m$ and event

label y_e, $p_m = y_e S(\mathbf{w}_1, \boldsymbol{\theta}_m) + (1 - y_e) S(\mathbf{w}_0, \boldsymbol{\theta}_m)$. $S(\mathbf{w}_i, \boldsymbol{\theta})$, $\forall i = 0, 1$, represents a sigmoid function,

$$S(\mathbf{w}_i, \boldsymbol{\theta}) = \frac{1}{1 + \exp\{-(w_{i0} + w_{i1}\theta_1 + \ldots + w_{iL}\theta_L)\}}. \tag{94}$$

The coefficients $\mathbf{w}_i = [w_{i0}, w_{i1}, \ldots, w_{iL}]$ capture how the community interact with messages related to true events and false events. Let $W = \{\mathbf{w}_i, \forall i = 1, 0\}$. We can write the probability distribution of r_m conditioned on $\boldsymbol{\theta}_m$ and y_e as

$$\begin{aligned} P(r_m | \boldsymbol{\theta}_m, y_e, W) = & \left[\left(S(\mathbf{w}_1, \boldsymbol{\theta}_m) \right)^{y_e} \left(S(\mathbf{w}_0, \boldsymbol{\theta}_m) \right)^{1 - y_e} \right]^{r_m} \\ & \times \left[\left(1 - S(\mathbf{w}_1, \boldsymbol{\theta}_m) \right)^{y_e} \left(1 - S(\mathbf{w}_0, \boldsymbol{\theta}_m) \right)^{1 - y_e} \right]^{1 - r_m}. \end{aligned} \tag{95}$$

The prior distributions for coefficients $\mathbf{w}_i$ are given by $\mathbf{w}_i \sim \text{Gaussian}(0, \lambda^{-1}\mathbf{I})$ for regularization.

The probabilistic graphical representation of the generative model is provided in Fig. 25. The notations are summarized in Table 15. We describe the generative process as follows:

1. Draw $\mathbf{w}_i \sim \text{Gaussian}(0, \lambda^{-1}\mathbf{I})$, $\forall i \in \{0, 1\}$.
2. For each event e:
   a) Draw its label $y_e \sim \text{Bernoulli}(\eta)$;
   b) For each message m related to event e:
      - Draw its feature $\boldsymbol{\theta}_m \sim f(\boldsymbol{\theta})$;
      - Draw its feedback $r_m \sim \text{Bernoulli}(p_m)$.

7.2.2 Model Learning

Let $E = \{e_1, e_2, \ldots, e_N\}$ represent the set of events. We denote $Y = \{y_e | e \in E\}$ as the set of event labels, $M = \cup_{e \in E} M_e$ as the set of relevant messages of all events, $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_m | m \in M\}$ as the set of features of all messages, $R = \{r_m | m \in M\}$ as

Table 15: The notations of the generative model.

N	number of events
N_e	number of relevant messages for event e
y_e	label of event e
$\boldsymbol{\theta}_m$	feature vector of message m
r_m	community feedback on message m
p_m	parameter of the Bernoulli distribution for r_m
$\mathbf{w}_i$	coefficient vector for sigmoid function S
λ, η	prior parameters for $\mathbf{w}_i$ and y_e, respectively

the set of community feedbacks on all messages, and $\Gamma = \{\eta, \lambda\}$ as the set of hyper parameters. The joint distribution of the observed data is given by

$$\begin{aligned} P(\boldsymbol{\Theta}, R, Y; W, \Gamma) &= P(R|\boldsymbol{\Theta}, W, Y) P(\boldsymbol{\Theta}|Y) P(Y|\Gamma) \\ &= \prod_{e \in E} \prod_{m \in M_e} P(r_m|\boldsymbol{\theta}_m, y_e, W) f_1(\boldsymbol{\theta}_m)^{y_e} f_0(\boldsymbol{\theta}_m)^{1-y_e} \\ &\quad \times \eta^{y_e} (1-\eta)^{1-y_e}. \end{aligned} \tag{96}$$

Since $f(\boldsymbol{\theta})$ is independent of other variables given y_e and $\boldsymbol{\theta}_m$, we can estimate $f(\boldsymbol{\theta})$ from the observed data of y_e and $\boldsymbol{\theta}_m$. We apply the non-parametric kernel density estimation method. Let $E_1 = \{e|y_e = 1, e \in E\}$ and $E_0 = \{e|y_e = 0, e \in E\}$. The estimated $f(\boldsymbol{\theta}|y_e)$ is given by

$$f(\boldsymbol{\theta}|y_e) \propto y_e \sum_{e \in E_1} \sum_{m \in M_e} K(\boldsymbol{\theta} - \boldsymbol{\theta}_m) + (1 - y_e) \sum_{e \in E_0} \sum_{m \in M_e} K(\boldsymbol{\theta} - \boldsymbol{\theta}_m), \tag{97}$$

where $K(\cdot)$ is the kernel function. There are various kernel functions can be used, e.g., Tophat and Gaussian kernels.

We next estimate the parameter W by maximizing the posterior probability given the observed data

$$\begin{aligned} L(W) &= \log P(W|\boldsymbol{\Theta}, R, Y, \Gamma) \\ &= \sum_{i \in \{0,1\}} \sum_{e \in E_i} \sum_{m \in M_e} r_m \log S(\mathbf{w}_i, \boldsymbol{\theta}_m) + \\ &\quad (1 - r_m) \log(1 - S(\mathbf{w}_i, \boldsymbol{\theta}_m)) - \frac{\lambda}{2} \mathbf{w}_i^\top \mathbf{w}_i + c_0 \end{aligned} \tag{98}$$

where c_0 is some constant. We employ the numerical Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [83] to solve for the unique optimal solution.

7.2.3 Online Prediction with Streaming Tweets

Let $M_e^{(h)}$ denote the observed tweets related to event e during a short period of T_h. The posterior probability of the event label is given by

$$P(y_e|M_e^{(h)}) \propto \prod_{m \in M_e^{(h)}} P(r_m|\boldsymbol{\theta}_m, y_e) f(\boldsymbol{\theta}_m|y_e) P(y_e|\eta). \tag{99}$$

At the beginning, we can initialize the parameter η for the prior distribution of y_e with any prior information. We then use the posterior to update the prior for y_e,

$$P(y_e|\eta^{(h)}) = P(y_e|M_e^{(1)}, M_e^{(2)}, \ldots, M_e^{(h)}). \tag{100}$$

As we continue to observe new streaming tweets $M_e^{(h+1)}$ for period T_{h+1}, we compute the new posterior for y_e as

$$P(y_e|M_e^{(1)}, M_e^{(2)}, \ldots, M_e^{(h)}, M_e^{(h+1)}) \propto \prod_{m \in M_e^{(h+1)}} P(r_m|\boldsymbol{\theta}_m, y_e) f(\boldsymbol{\theta}_m|y_e) P(y_e|\eta^{(h)}). \tag{101}$$

Therefore, the online streaming prediction algorithm does not need to store or re-process the tweets observed in the past. After each period T_h, we can update the prediction of event label as $\arg\max P(y_e|M_e^{(1)}, M_e^{(2)}, \ldots, M_e^{(h)})$.

7.3 Experiments

We run experiments on a dataset of tweets collected from Twitter.com, and evaluate the event credibility prediction performance of the proposed model.

7.3.1 Datasets

7.3.1.1 Collection

A set of events and trending topics were collected from facebook.com, twitter.com, and snopes.com on a daily basis. Note that snopes.com also provides labels for the

listed events. For other unlabelled events, we asked 5 annotators to label the events as true or false. They consulted external resources including major news websites such as cnn.com and nytimes.com to determine the credibility of events. To reduce uncertainty in event labelling, we only included the events whose labels were agreed upon by at least 4 annotators.

For each event, we gathered tweets relevant to the event from twitter.com via Twitter Search API[1]. We constructed a query for each event with a group of keywords, and sent the search request to twitter.com to retrieve tweets contain all keywords in the query. Each returned tweet contains the tweet text, the author information, and the number of "Retweets" and "Favorites". In the experiments, we define a tweet receives positive feedback if the total count of "Reweets" and "Favorites" is greater than 1 and negative feedback vice versa. The Twitter Search API only returns tweets from the past week. In addition, to avoid collecting duplicated tweets, we removed all retweets.

We limited the maximum number of collected tweets to 500 per event, and also discarded events with less than 30 tweets. We continued this process for a period of three months from November 2014 to January 2015 until collected 104 events consisting of 52 true events and 52 false events. The total number of collected relevant tweets is 29,345.

7.3.1.2 Features

For each collected tweet, we extract a set of features similar to those discovered in [26], including the author-based features: registration age, number of posted tweets, number of followers, number of friends, and friend-follower ratio; the content-based features: sentiment score, subjectivity score, contains URLs, contains user mentions, contains question marks, and contains first pronouns. Note that the friend-follower

[1] `https://dev.twitter.com/rest/public/search`

Table 16: Aggregation features for event classification.

Author	Average registration age
	Average number of posted tweets
	Average number of followers
	Average number of friends
Content	Fraction of messages contain URLs
	Average sentiment score
	Fraction of messages with positive sentiment
	Fraction of messages with negative sentiment
	Fraction of messages contain user mentions
	Fraction of messages contain question marks
	Fraction of messages contain first pronouns
	Fraction of messages with positive feedback

ratio is computed as (number of friends + 100)/(number of followers + 100), so as to smooth the ratio when the number of followers is very small.

The raw values of author features vary over a very wide range. We apply the transformation $F(f_a) = \frac{f_a}{f_a+d_f}$ to author feature f_a, where d_f is some constant for each feature. We set d_f as the median of observed f_a values. The transformed feature values are now in the range of $[0, 1]$.

7.3.2 Batch Prediction Performance

We first evaluate the offline batch prediction performance of the proposed algorithm, where all collected relevant tweets are used for prediction. We set the prior parameter for regularization of W as $\lambda = 1$, and use Tophat kernel with a bandwidth of 0.4 for the nonparametric kernel estimator in $f(\boldsymbol{\theta})$.

7.3.2.1 Baselines

The baseline algorithms are the classification algorithms using the aggregation features of events as proposed in [26]. We apply the Decision Tree and linear Support Vector Machine (SVM) methods for this purpose. For those algorithms, we use the aggregation features presented in Table 16 including the author-based features and the content-based features, which were shown to have good discrimination power. We

Table 17: Batch prediction performance for the true and false event classes.

Algorithm	All	True Event Class			False Event Class		
	Accuracy	Precision	Recall	F_1 score	Precision	Recall	F_1 score
Proposed	0.823	0.829	0.826	0.822	0.838	0.820	0.821
SVM	0.771	0.807	0.760	0.763	0.766	0.783	0.765
Decision Tree	0.725	0.735	0.740	0.715	0.764	0.710	0.720

omit some propagation-based aggregation features, e.g., the maximum size of a level in the propagation tree, because they require the complete set of tweets relevant to an event in order to reconstruct the tweet propagation tree, but our dataset does not guarantee such completeness. Also, generating those features often causes much longer delay since the propagation of tweets takes time.

7.3.2.2 Performance Comparison

We run 10-fold cross validation and report the average results. Table 17 shows the overall classification accuracy as well as the precision and recall performance of both the true event and false event classes. The proposed model achieves an overall prediction accuracy of 82.3%, and it outperforms the baseline methods across all metrics. Therefore, modelling events at the individual message level is a more effective approach to event credibility prediction when only a subset of tweets related to events are available.

7.3.3 Online Streaming Prediction

We investigate the online prediction performance of the proposed model with streaming tweets, where the probability of the event label is updated online through (101). In Fig. 26, we show the online prediction accuracy versus the number of observed tweets. The performance first improves significantly as the number of tweets increases. The algorithm achieves a quite good accuracy of 78.3% even with only 200 tweets. After the number of tweets reaches around 200 to 300, the performance improvement slows down. This is because after the number of past tweets becomes

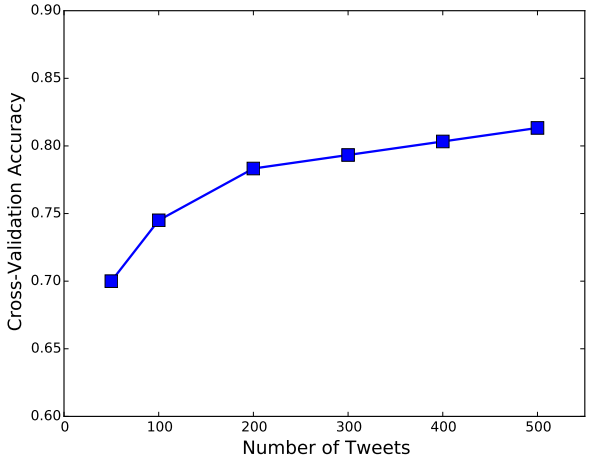


Figure 26: Online prediction performance with streaming tweets.

large, a relatively small number of new streaming tweets can only slightly influence the prediction. Also, as more and more tweets are observed, the accuracy converges to the limit of the prediction performance of the model. The results confirm that the online prediction algorithm quickly approaches the prediction accuracy of the batch prediction with only a few hundred tweets.

7.3.4 Impact of Parameters and Kernels

In the following, we investigate the impact of parameters and density estimation kernels on the proposed algorithm based on the 10-fold cross-validation results.

7.3.4.1 Impact of Parameter λ

We examine the impact of parameter λ on the proposed model. In Fig. 27, we show the cross-validation classification accuracy with λ taking different values. We fixed the density estimation kernel as Tophat with its bandwidth set to 0.4. We can see that setting λ around 1 achieves the best accuracy performance. As λ becomes too large, the accuracy decreases, because the excessive regularization on W causes underfitting.

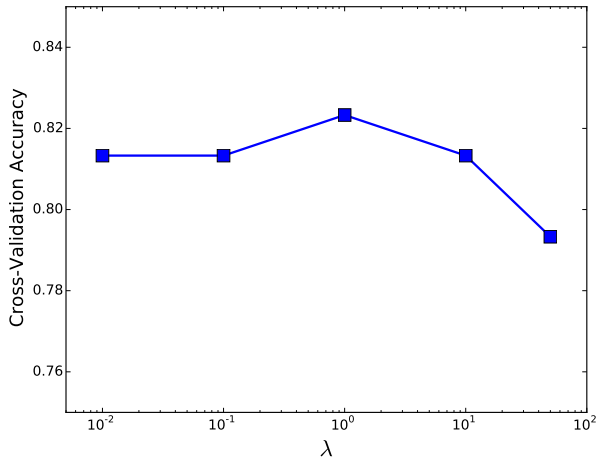


Figure 27: Impact of parameter λ.

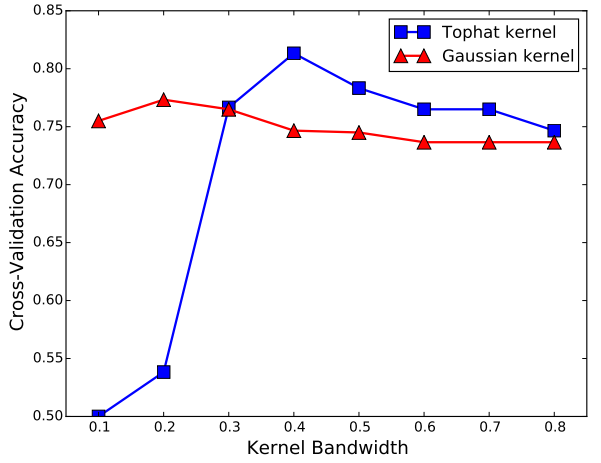


Figure 28: Impact of density estimation kernels.

7.3.4.2 Impact of Density Estimation Kernel

In Fig. 28, we show the accuracy for various density estimation kernels used in (97), including Tophat and Gaussian kernels with different bandwidths, while fixing λ as 1. For the Tophat kernel, setting the bandwidth to 0.4 achieves the best performance, while for the Gaussian kernel, a bandwidth of 0.2 works best. The best accuracy performance of the Tophat kernel is slightly better than that of the Gaussian kernel. When the bandwidth is too large, the performance of both kernels degrades due to

over-smoothing in the estimated density function. Meanwhile, for the Tophat kernel, when the bandwidth is too small, its prediction performance dramatically drops due to under-smoothing. Overall, the Gaussian kernel is less sensitive to the bandwidth than the Tophat kernel, because the Gaussian kernel is smoother than the Tophat kernel.

7.4 Conclusion

In this chapter, we developed a probabilistic generative model for real-time event credibility prediction in Twitter-like social media. In contrast to aggregation analysis techniques that require a complete set of tweets related to events, our model only needs signals extracted from individual tweets. Using the generative model, we proposed an online streaming prediction algorithm without storing or reprocessing the past tweets. Through experiments on the Twitter dataset, we showed that the batch prediction performance of our model outperforms other algorithms based on aggregation analysis, and the online streaming prediction performance quickly approaches that of the batch prediction with only a few hundred tweets.

CHAPTER VIII

USER TRUSTWORTHINESS PREDICTION VIA PMRF

8.1 Introduction and Related Works

Online social networks e.g., Facebook and Twitter, are becoming important platforms for making social connections and sharing information. However, due to the open nature of such social networking platforms, they are inherently vulnerable to malicious users or spammers who misuse social networks to perform malevolent activities [21, 48] such as spamming, phishing, and spreading computer viruses. Hence, user trustworthiness in social networks has attracted wide interest from both government and industry [103]. Existing works in [106, 117, 60] proposed to predict user trustworthiness based on user behavior patterns in social networks. They introduced various features extracted from user profiles and user generated social content, and used them to detect spammers using classification tools from machine learning. Although such feature-based classification approaches can capture behavioural characteristics of individual users, they ignore the relationships between users.

In most online social networks, users can connect to other people they know or share common interests. The establishment of bidirectional connections between users often indicates some degree of similarity in their trustworthiness. In some social networks like Facebook, the connecting request initialized by one user requires explicit approval from the other user before they become connected, and in other social networks like Twitter, although users can unilaterally establish directed connections, e.g., following other people, bidirectional connections are mostly established between friends who know each other in real life, or between people who share mutual interests.

Hence, users with close social relationships are more likely to have similar roles in social networks. For example, the study on the Twitter social network has revealed that criminal user accounts tend to be socially connected to form a small-world network [118].

A number of other works utilized the connection patterns in social network structure. In [23] the authors evaluated user trustworthiness based on the density of interconnections between users, assigning higher trustworthiness to users with higher degree of connections. In [15], the authors proposed to use social network characteristics of community formation to learn classification models for identifying spammers. However, in today's social networks, there can be a significantly larger number of malicious users and they can connect to each other to increase their social connections. Indeed, the authors in [118] examined the Twitter network as a particular case and found that criminal accounts tend to be socially connected to form a small-world network.

In [126], a socially regularized matrix factorization model was used to learn latent user features from social activities for spammer detection in social networks. By exploiting users' social relationships, it imposed a social regularization term on matrix factorization. The latent factor vector of a user should be similar to their connected users', since they share similar interests and may perform similar social activities. In [46], the authors employed social relationships to regularize the least squares model for spammer classification, where an extra penalty is added to the loss function during training if two users with close social relationships have different predicted labels. However, the user's social relationships are not utilized to predict the label of an unknown target user. In [101], the authors proposed a social network aided Bayesian spam email filter, which adjusts the keyword weights based on the social closeness between the email receiver and sender, as users with high closeness are less likely to send spam emails to each other.

In this chapter, we propose a probabilistic model based on Pairwise Markov Random Field (PMRF) that takes into account both user features and social relationships [133]. We use PMRF to express a proper factorization of the joint distribution of users based on their social relationships. As PMRF can be conveniently represented by a probabilistic graph consisting of edges and nodes, we can apply the Belief Propagation (BP) [87] algorithm to exploit the graph structure to perform inference efficiently, and hence, the complexity of the algorithm grows only linear in the number of users. In the experiment, we apply the proposed algorithm to predict spammers in the Twitter datasets, and show that the proposed PMRF model can effectively exploit the social relationships to significantly improve the prediction performance.

8.2 User Feature-Based Methods

In this section, we briefly introduce how user features are derived and used to predict user trustworthiness in social networks. As the research results in [60, 117] show, the different patterns of trusted users and untrusted users can be captured by carefully designed user features. Generally, the user features are mainly extracted from the following two major categories of social data:

8.2.0.3 User Profiles

The profiles of trusted users tend to be more complete and verifiable than those of untrusted users. For example, trusted users usually provide more details of personal information, such as his professional title and employer, and they may also provide links to personal webpages, so that other people can actually verify the user's true identity in real-life. In addition, the existing time of a trusted user' account since it was first registered is usually longer, whereas many untrusted users are newly registered users.

8.2.0.4 User Content

Users generate data and disseminate information in online social networks, e.g., post tweets in Twitter. We can directly look into user generated content to determine if a user is behaving improperly, e.g., posting spam messages that include phishing URLs. Also, we can extract certain patterns of users' posting behaviors, such as the number of duplicated messages users post. The specific features that are effective depend on the form of the social network, and need to be designed accordingly.

The basic approach to analyzing user trustworthiness is using the classification machine learning tools [106, 117, 60], e.g., Support Vector Machine and Decision Tree. The classifiers are first trained with labelled user data, and given a unknown user they generate classification results as the predicted user trustworthiness. However, such classifiers treat each individual user separately from others in social networks, and ignore the social relationships between users.

8.3 Modelling Trustworthiness on PMRF

In order to exploit the social relationships to improve user trustworthiness prediction, we formulate the problem as a probabilistic inference problem using a PMRF graphical model, where the social relationships can be conveniently modeled as edges between hidden nodes. In addition, the inference in PMRF can be carried out efficiently by using the BP algorithm.

8.3.1 Probabilistic Problem Formulation

We assume a set $\mathbb{U}$ of M users in the social network, $\mathbb{U} = \{1, \ldots, M\}$. Our goal is to infer the trustworthiness r_u of each user $u \in \mathbb{U}$. We model r_u as a discrete random variable, which takes values from a discrete set $\Gamma = \{s_1, s_2, \ldots, s_L\}$, $|\Gamma| = L$. For example, with $\Gamma = \{0, 1\}$, we can use 0 and 1 to represent "malicious/untrusted" and "normal/trusted", respectively. For user u, we also represent the user features

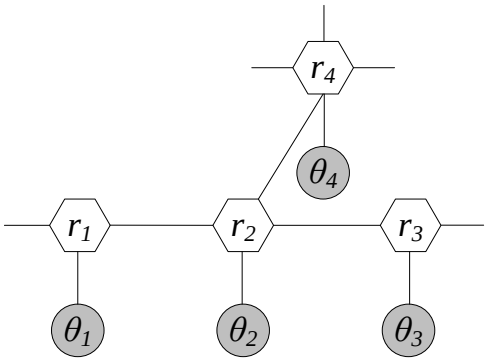


Figure 29: Illustration of the PMRF model for user trustworthiness.

by vector $\boldsymbol{\theta}_u = [\theta_1, \theta_2, \ldots, \theta_p]$ with length p, which can be extracted from the social data as discussed in Sec. 8.2. Let $\mathbb{R} = \{r_1, \ldots, r_M\}$ denote the set of trustworthiness variables, and $\Theta = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M\}$ denote the set of user features for all users.

To take into account the dependency between users due to social relationships, we need to jointly model all variables in $\mathbb{R}$. Let $P(\mathbb{R}|\Theta)$ be the joint posterior distribution of $\mathbb{R}$, given the observed user features in Θ. Then to compute r_u, we infer the marginal posterior distribution $P(r_u|\Theta)$ from $P(\mathbb{R}|\Theta)$. The direct computation can be expressed as follows

$$P(r_u|\Theta) = \sum_{r_1 \in \Gamma} \cdots \sum_{r_{u-1} \in \Gamma} \sum_{r_{u+1} \in \Gamma} \cdots \sum_{r_M \in \Gamma} P(\mathbb{R}|\Theta), \tag{102}$$

which is summation over all variables except r_u. Obviously, the computational complexity grows exponentially as $\mathcal{O}(L^M)$. Considering the large number of users in social networks, (102) is computationally infeasible. In the following, we propose a PMRF model that properly factorizes the joint distribution $P(\mathbb{R}|\Theta)$ into local functions according to social relationships between users, and hence we can apply the BP algorithm to infer $P(r_u|\Theta)$ efficiently.

8.3.2 Modelling via PMRF

A PMRF consists of hidden and observed nodes [122], in which the statistical dependencies between two connected hidden nodes are represented by compatibility functions, and the dependency relations between the observed nodes and hidden nodes

are represented by local evidence functions. We model the joint posterior probability distribution $P(\mathbb{R}|\Theta)$ in a PMRF $\mathcal{G}$ as illustrated in Fig. 29. For each user u, whose trustworthiness r_u needs to be predicted, we assign a hidden node (or variable node), shown as a hexagon, and we further connect it to an observed node (or feature node), shown as a circle, which represents the observed features $\boldsymbol{\theta}_u$ of user u. To capture the dependence among variables induced by social relationships, variable nodes r_u and r_v are connected via an edge, if there exists a social relationship between users u and v. Let $\mathcal{E}$ denote the set of all edges between variable nodes. We can express a proper factorization of $P(\mathbb{R}|\Theta)$ represented by PMRF $\mathcal{G}$ as follows

$$P(\mathbb{R}|\Theta) = \frac{1}{Z} \prod_{(u,v)\in\mathcal{E}} \psi_{uv}(r_u, r_v) \prod_{u=1}^{M} \phi_u(r_u|\boldsymbol{\theta}_u) \tag{103}$$

where Z is a normalization factor, $\psi_{uv}(r_u, r_v)$ is a compatibility function between r_u and r_v, and $\phi_u(r_u|\boldsymbol{\theta}_u)$ is the local evidence function for r_u given $\boldsymbol{\theta}_u$.

Firstly, we specify the local evidence function $\phi_u(r_u|\boldsymbol{\theta}_u)$, which describes the probability distribution of r_u given the observed user feature $\boldsymbol{\theta}_u$. We can use the output probability distribution from the probabilistic classifiers. As one example, we use the Logistic Regression [18] to classify users as spammers ($r_u = 0$) or normal users ($r_u = 1$), in which the probability of $r_u = 1$ given the observed feature vector $\boldsymbol{\theta}_u$ can be computed as follows

$$P_{\text{reg}}(r_u = 1|\boldsymbol{\theta}_u) = \frac{1}{1 + \exp\{-(c_0 + c_1\theta_1 + \ldots + c_p\theta_p)\}}, \tag{104}$$

where $\mathbf{c} = [c_0, c_1, \ldots, c_p]$ is a logistic regression coefficient vector to be learnt from training datasets, and

$$P_{\text{reg}}(r_u = 0|\boldsymbol{\theta}_u) = 1 - P_{\text{reg}}(r_u = 1|\boldsymbol{\theta}_u). \tag{105}$$

Hence, we can let $\phi_u(r_u|\boldsymbol{\theta}_u) = P_{\text{reg}}(r_u|\boldsymbol{\theta}_u)$.

The compatibility function $\psi_{uv}(r_u, r_v)$ needs to be defined properly to reflect the influence users u and v have on each other in the social network. Basically, r_u and r_v

should be compatible and have similar values if there exists a strong social relationship between users u and v. In this work, we define $\psi_{uv}(r_u, r_v)$ as

$$\psi_{uv}(r_u, r_v) \propto \exp\left\{-\alpha(r_u - r_v)^2\right\}, \tag{106}$$

where $\alpha > 0$ is some parameter that adjusts the influence of one user on his connected users. Since r_u only takes discrete values from Γ, $\psi_{uv}(r_u, r_v)$ can also be explicitly expressed as

$$\psi_{uv}(r_u = s_i, r_v = s_j) = q_{ij}, \tag{107}$$

where $0 \leq q_{ij} \leq 1$ and $\sum_{1 \leq j \leq L} q_{ij} = 1$. We can interpret q_{ij} as the probability of $r_v = s_j$ given $r_u = s_i$. A larger q_{ii} means users u and v are more likely to have similar trustworthiness, i.e., $r_u = r_v = s_i$.

8.3.3 Inference Using BP

The PMRF model expresses the factorization of $P(\mathbb{R}|\Theta)$ into many local functions as shown in (103), and hence, we apply the BP algorithm to exploit such structure to efficiently infer the marginal probability distribution $P(r_u|\Theta)$, $\forall u \in \mathbb{U}$. This is one important computational advantage of PMRF models.

8.3.3.1 BP Algorithm

BP is a message-passing algorithm that operates on probabilistic graphs, where messages are exchanged between nodes along edges. By exploiting the graph structure, BP efficiently computes marginal functions from complex global functions of large number of variables. When the graph has a tree structure, BP computes the exact results. Even in graphs with loops, we can obtain very good approximate results by applying the loopy BP algorithm [123], in which the probabilistic messages are iteratively exchanged between variable nodes until convergence. In this work, since users in social networks can easily connect to each other to form loops, the proposed

PMRF model will have loops in many cases, and hence, we need to apply the iterative loopy BP algorithm.

During each iteration n, at any variable node b, we update the messages sent to its neighbors. To compute $m_{b,a}^{(n)}(r_a)$, the message sent from variable node b to neighbor node a, we compute the product of all incoming messages in last iteration from its neighbors except the one from node a, and multiply it with the local evidence function and the compatibility function between nodes a and b. This message is given as follows

$$m_{b,a}^{(n)}(r_a) \propto \sum_{r_b \in \Gamma} \psi_{ab}(r_a, r_b)\phi_b(r_b|\boldsymbol{\theta}_b) \prod_{c \in \mathcal{N}(b)\setminus a} m_{c,b}^{(n-1)}(r_b), \tag{108}$$

where $\mathcal{N}(b)$ denotes the set of users who are connected to user b, whose corresponding variable nodes are connected to r_b in PMRF.

After BP converges, the marginal distribution $P(r_u|\Theta)$ of r_u can be computed according to

$$P(r_u|\Theta) \propto \phi_u(r_u|\boldsymbol{\theta}_u) \prod_{v \in \mathcal{N}(u)} m_{v,u}^{(n)}(r_u). \tag{109}$$

The predicted trustworthiness $\hat{r}_u$ for user u is given by

$$\hat{r}_u = \text{argmax}_{r_u \in \Gamma} P(r_u|\Theta). \tag{110}$$

We summarize the BP algorithm in Algorithm 4.

8.3.3.2 Complexity Analysis

The complexity of updating a message using (108) is $\mathcal{O}(|\Gamma|K)$, where K is the average degree of each variable node on PMRF. In each iteration, each variable node updates and sends out K messages to its neighbor nodes, and the total number of updated messages is $\mathcal{O}(MK)$. Hence, the overall complexity in terms of multiplications is $\mathcal{O}(|\Gamma|MK^2)$. Note that the proposed algorithm converges quickly, on the average in 10 iterations. Hence, the complexity of the algorithm grows only linear in the number of users.

Algorithm 4 BP Algorithm on PMRF

- Initialization. Initialize $m_{a,b}(r_b)$ as $m_{a,b}^{(0)}(r_b) = \frac{1}{|\Gamma|}$, and set iteration counter $n = 1$.
- Iterative message-passing until convergence.
  (1) In iteration n, at each node b, update $m_{a,b}^{(n)}(r_b)$ for its neighors $\mathcal{N}(b)$ using Eqn. (108);
  (2) $n = n + 1$, and repeat step (1) until convergence.
- Compute the marginal probabilities $P(r_u|\Theta)$ using Eqn. (109);
- Compute the trustworthiness of all users using Eqn. (110).

8.4 Experimental Evaluation

8.4.1 Datasets

We use the Twitter dataset prepared by [117], in which the authors crawled the real Twitter user profiles and identified spammers that post URL links of malicious websites, e.g., phishing websites. The dataset includes 1,000 spammers and 10,000 non-spammers (normal users). The crawled data for each user contain the basic user profile information, e.g., account creation time, the list of followers and followees, and the 40 most recent Tweets. From the user data we can extract various user features such as those introduced in [60, 117]. In our experiment, we use the following three user features in Table 18 that are reported to be very effective in discriminating spammers from normal users. We also extract user relationships using the bidirectional following connections, i.e., user pairs that follow each other in the Twitter dataset.

Table 18: User features in Twitter dataset

Feature	Definition
Account age	How long ago the user account was created.
URL rate	The average number of posted URLs per tweet.
Reply ratio	The ratio of the number of tweets that are replies to other users to the total number of all tweets.

Table 19: Classification performance comparison for the spammer and non-spammer classes.

Algorithm	All	Spammer Class			Non-Spammer Class		
	Accuracy	Precision	Recall	F_1 score	Precision	Recall	F_1 score
Proposed PMRF	0.895	0.891	0.900	0.896	0.899	0.890	0.895
SVM	0.735	0.730	0.745	0.738	0.740	0.725	0.732
Logistic Regression	0.733	0.734	0.730	0.732	0.731	0.735	0.733
Naive Bayes	0.725	0.686	0.830	0.751	0.785	0.620	0.693

8.4.2 Performance Evaluation

In the experiment, we predict the trustworthiness of users by classifying the users into spammers and non-spammers. To evaluate classification performance, we first create balanced training data from the dataset. We include all 1,000 spammers of the original dataset, and randomly sample 1,000 non-spammers. We measure the performance in terms of overall accuracy as well as the precision, recall, and F_1-score metrics for both spammer and non-spammer classes. For all introduced metrics, the higher the value the better the performance. However, the individual metric of precision or recall does not reflect the performance of the algorithms very well by itself, since high precision may be achieved at low recall, and high recall may be achieved at low precision, while the F_1-score that calculates the harmonic mean of precision and recall is a more balanced metric.

In the PMRF model, we let $r_u = 0$ to indicate that user u is a spammer and $r_u = 1$ for non-spammer. The compatibility function in (107) is specified by the transition probability $q_{ii} = 0.95$, $\forall i = 0, 1$, and $q_{ij} = 0.05$ for $i \neq j$. In addition, we use the probabilities computed by the Logistic Regression classifier (104) as $\phi_u(r_u|\boldsymbol{\theta}_u)$. We compare the performance of the proposed PMRF-based algorithm with the following classification algorithms based only on users features, including:

- **Support Vector Machine (SVM)**: We use the `svmtrain()` function from the Statistics and Machine Learning Toolbox in Matlab to train a linear support vector machine classifier;

- **Logistic Regression**: To learn the logistic regression coefficients, we use the multinoimal logistic regression function `mnfit()`;
- **Naive Bayes**: We train the Naive Bayes classifier using the `NaiveBayes.fit()` function.

We split the data such that 80% of users in each class are used for training and the rest 20% for testing. We summarize the results on the testing data in Table 19. For all algorithms, we use the same features shown in Table 18. The results show that our PMRF-based algorithm outperforms other classification algorithms significantly, and it consistently achieves better performance in all metrics, improving classification performance for both spammer and non-spammer classes. This confirms that online social relationships between users can be utilized for user classification, and our proposed algorithm effectively exploits such relationships.

For all other classification algorithms, their accuracy results are quite close. Among them, the SVM classifier has the best accuracy, yet 18% worse than the proposed PMRF algorithm. Noticeably, all of them classify users individually. They take each user's features as input and predict his/her label, disregarding the labels of other users. This suggests that it is difficult to gain further performance improvement without taking user relationships into account.

8.4.2.1 Impact of Parameters

We investigate the impact of the compatibility function (107) on classification performance. In the experiment, we always let $q_{00} = q_{11}$, so that the effects of the compatibility function on both spammer and non-spammer classes are identical. In Fig. 30, we show the overall accuracy of the proposed algorithm with q_{ii} increasing from 0.5 to 1. We can see that when $q_{ii} = 0.5$, the proposed algorithm has the same performance as that of Logistic Regression, since setting $q_{ii} = 0.5$ means no compatibility is imposed on the labels of two connected users, i.e., given the label of a

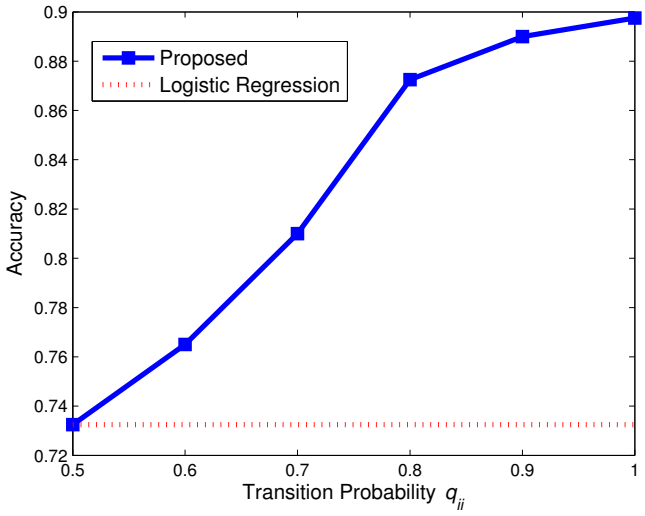


Figure 30: Impact of the compatibility function on accuracy.

user, the other connected user can have either label, spammer or non-spammer, with equal probabilities. As q_{ii} increases, the accuracy first improves significantly, but then the performance begins to saturate when q_{ii} is large enough. This is due to several factors, e.g., some users are not connected to other users and thus changing compatibility function does not effect their predicted labels, or some users' probabilities of labels are strongly dominated by their individual features.

We note that even though in this experiment setting $q_{ii} = 1$ seems to provide the best performance, in general the appropriate value of q_{ii} should be chosen based on the cross-validation results on the datasets. Choosing a moderate value can reduce the influence of a single user, and hence only when there are large enough number of users with identical labels socially connected to a user, his/her label will be significantly influenced. This can be useful for some scenarios, e.g., where spammers have tricked some new non-spammers to connect to them. Also, when the local evidence $\phi_u(r_u|\boldsymbol{\theta}_u)$ computed based on the user feature has low prediction accuracy, a smaller q_{ii} (greater than 0.5) can reduce false label propagation compared to $q_{ii} = 1$.

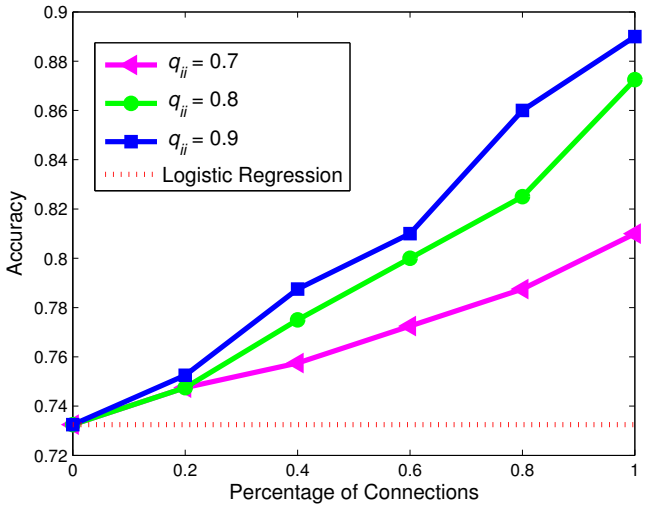


Figure 31: Impact of social relationships on accuracy.

8.4.2.2 Impact of Social Relationships

We next examine how the richness of social connections impacts the proposed algorithm. From the original dataset, we randomly sample a proportion of the social connections between users to generate the required datasets for our experiment. In Fig. 31, we show the accuracy results for varying percentage of social connections (relative to the original dataset), under $q_{ii} = \{0.7, 0.8, 0.9\}$. As the percentage of social connections decreases, the accuracy of the proposed algorithm decreases. When no social relationships can be utilized, it degenerates to Logistic Regression that classifies users individually. Comparing $q_{ii} = 0.9$ to $q_{ii} = 0.7$, we also find that the performance for larger q_{ii} is more sensitive to the richness of social connections.

8.5 Conclusion

In this chapter, we proposed a probabilistic graphical model based on PMRF for predicting user trustworthiness in online social networks, which incorporates both user features and social relationships. We applied the BP algorithm to exploit the graph structure to perform inference efficiently. The computational complexity is linear in the number of users. In the experiment on the Twitter dataset, we applied the

proposed algorithm to predict spammers, and the results showed that it outperforms other user feature-based classification algorithms in terms of both accuracy and F_1 score. Hence, the proposed algorithm effectively leveraged online social relationships to improve prediction performance.

CHAPTER IX

CONCLUSION

9.1 Summary of Contributions

In this dissertation, we addressed the challenging social computing problems in recommender systems and social media information mining which have a large number of variables as well as complex dependency relationships between those variables. We constructed proper probabilistic graphical models suitable for the problems and developed computationally efficient algorithms. The contributions are summarized as follows:

- We proposed probabilistic graphical models for Collaborative Filtering (CF) recommender systems. We computed user similarity for the neighborhood CF method using factor graphs. Then we solved the recommender system problem by model unknown item ratings on Pairwise Markov Random Fields (PMRFs) incorporating both user similarity and item similarity. Experimental evaluation results using the 100K MovieLens dataset showed that the proposed algorithms outperformed other neighborhood methods.
- We proposed a Bayesian network model for social recommendation systems, which was constructed based on relations between users in social networks. The experimental results on the Epinions dataset showed that the proposed algorithm achieved better top-N recommendation performance in terms of recall than both the random walk based and voting based social recommendation algorithms.
- We proposed a semi-distributed item-based CF approach to privacy-preserving recommender systems. The architecture was based on factor graphs for computing

item similarities. Through information-theoretic analysis, we showed that the proposed algorithm effectively preserves user privacy. The experimental results on the MovieLens dataset demonstrated that the algorithm also achieved superior rating prediction performance.

- We proposed a factor graph model for jointly detecting spammers in shilling attacks on CF recommender systems, which exploited the colluding spammers' collaborative spamming behaviors towards target items. We showed through experiments that the proposed algorithm significantly improved detection accuracy as the number of target items increases.
- We proposed a rank aggregation approach and a popularity-biased Bayesian personalized ranking approach to exploiting both popularity and similarity for social link recommendation. Experimental evaluation results on real-world datasets showed that the proposed algorithms can combine the advantages of popularity-based and similarity-based algorithms to improve AUC and precision.
- We proposed a probabilistic generative model for real-time event credibility prediction in Twitter-like social media using streaming tweets. Through experiments on the Twitter dataset, we showed that the batch prediction performance of our model outperformed other algorithms based on aggregation analysis, and the online streaming prediction performance quickly approached that of the batch prediction with only a few hundred tweets.
- We proposed a probabilistic PMRF graphical model for predicting user trustworthiness in online social networks, which incorporated both user features and social relationships. In the experiment on the Twitter dataset, we applied the proposed algorithm to predict spammers, and the results showed that it outperformed other classification algorithms using only features of individual users.

9.2 Future Research

In this dissertation, the proposed probabilistic graphical models for recommender systems do not incorporate context information, e.g., time of the day and day of the week. In many online activities like music listening and movie watching, contexts can affect user preferences on items. Future research should extend graphical models to context-aware recommendation by using suitable models like conditional Markov random fields. In addition, the proposed recommender system models are for predicting item ratings, and are evaluated using metrics like root mean squared error. However, the accuracy of rating prediction does not reflect the relative rankings of items as seen by a user. Future research should consider ranking metrics for graphical models.

In detection of shilling attacks on recommender systems, the proposed factor graph model achieves good performance against the Average attacks. In practice, spammers may launch more complicated shilling attacks that do not follow a simple strategy, which can be a combination of both push and nuke attacks. Future research needs to design more robust detection algorithms to deal with more sophisticated attacks.

In social event credibility prediction, we use all observed tweets starting from the beginning, but ignore the temporal dynamics in information credibility. However, the tweets appear at the very beginning may be less reliable, whereas later tweets are likely more accurate as more evidences become available. Future research needs to take into account such temporal effects to improve the online credibility prediction algorithm.

In social media user trustworthiness prediction, social connections are modeled as edges in PMRF models without considering the strength of those connections. Future research should properly model the connection strength as users may interact more frequently with some close connections than others. Also, network structure analysis techniques such as PageRank [84] and SALSA [61] can be used to measure the importance of connections.

REFERENCES

[1] Ackerman, M. S., Cranor, L. F., and Reagle, J., "Privacy in e-commerce: Examining user scenarios and privacy preferences," in *Proceedings of the 1st ACM Conference on Electronic Commerce*, (Denver, CO, USA), pp. 1–8, 1999.

[2] Adomavicius, G. and Tuzhilin, A., "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 734–749, Jun. 2005.

[3] Adomavicius, G. and Tuzhilin, A., "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. on Knowl. Data Eng.*, vol. 17, pp. 734–749, Jun. 2005.

[4] Amatriain, X., "Beyond data: from user information to business value through personalized recommendations and consumer science," in *Proc. CIKM'13*, pp. 2201–2208, 2013.

[5] Ayday, E. and Fekri, F., "Belief propagation based iterative trust and reputation management," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, pp. 375–386, May/June 2012.

[6] Ayday, E., Zou, J., Einolghozati, A., and Fekri, F., "A recommender system based on belief propagation over pairwise Markov random fields," in *Proc. the 50th Annual Allerton Conference on Communication, Control, and Computing*, 2012.

[7] Ayday, E., Einolghozati, A., and Fekri, F., "BPRS: Belief propagation based iterative recommender system," in *Proceedings of the 2012 IEEE International Symposium on Information Theory*, (Cambridge, MA), pp. 1992–1996, 2012.

[8] Ayday, E. and Fekri, F., "A belief propagation based recommender system for online services," in *Proc. 4th ACM Conference on Recommender Systems (RecSys)*, pp. 217–220, 2010.

[9] Ayday, E. and Fekri, F., "BP-ITRM: belief propagation for iterative trust and reputation management," in *Proceedings of IEEE International Symposium on Information Theory (ISIT'11)*, (Saint Petersburg, Russia), 2011.

[10] Ayday, E. and Fekri, F., "BP-P2P: Belief propagation-based trust and reputation management for P2P networks," in *Proceedings of the 9th Annual IEEE*

Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pp. 578–586, 2012.

[11] Balabanovic, M. and Shoham, Y., "Fab: Content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, pp. 66–72, 1997.

[12] Barber, D., *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[13] Basu, A., Vaidya, J., Kikuchi, H., and Dimitrakos, T., "Privacy-preserving collaborative filtering on the cloud and practical implementation experiences," in *Proc. CLOUD'13*, pp. 406–413, 2013.

[14] Berkovsky, S., Eytani, Y., Kuflik, T., and Ricci, F., "Enhancing privacy and preserving accuracy of a distributed collaborative filtering," in *Proceedings of the 2007 ACM Conference on Recommender Systems*, pp. 9–16, 2007.

[15] Bhat, S. Y. and Abulaish, M., "Community-based features for identifying spammers in online social networks," in *Proc. ASONAM'13*, pp. 100–107, 2013.

[16] Bian, J., Yang, Y., and Chua, T.-S., "Predicting trending messages and diffusion participants in microblogging network," in *SIGIR'14*, pp. 531–540, 2014.

[17] Billsus, D. and Pazzani, M. J., "Learning collaborative information filters," *Proc. ICML'98*, pp. 46–54, 1998.

[18] Bishop, C. M., *Pattern Recognition and Machine Learning*. Springer, 2006.

[19] Breese, J. S., Heckerman, D., and Kadie, C., "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. UAI*, 1998.

[20] Breese, J. S., Heckerman, D., and Kadie, C., "Empirical analysis of predictive algorithms for collaborative filtering," *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43–52, 1998.

[21] Brown, G., Howe, T., Ihbe, M., Prakash, A., and Borders, K., "Social networks and context-aware spam," in *Proc. CSCW'08*, pp. 403–412, 2008.

[22] Burke, R., Mobasher, B., Williams, C., and Bhaumik, R., "Classification features for attack detection in collaborative recommender systems," in *Proc. KDD'06*, pp. 542–547, 2006.

[23] Buskens, V., "The social structure of trust," *Social Networks*, vol. 20, no. 3, pp. 265–289, 1998.

[24] Canny, J. F., "Collaborative filtering with privacy," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pp. 45–57, 2002.

[25] Cao, N., Wang, C., Li, M., Ren, K., and Lou, W., "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 222–233, Jan. 2014.

[26] Castillo, C., Mendoza, M., and Poblete, B., "Information credibility on twitter," in *WWW'11*, pp. 675–684, 2011.

[27] Cha, M., Haddadi, H., Benevenuto, F., and Gummadi, K. P., "Measuring user influence in twitter: The million follower fallacy," in *ICWSM'10*, pp. 10–17, 2010.

[28] Chirita, P.-A., Nejdl, W., and Zamfir, C., "Preventing shilling attacks in online recommender systems," in *Proc. WIDM'05*, pp. 67–74, 2005.

[29] Defazio, A. J. and Caetano, T. S., "A graphical model formulation of collaborative filtering neighbourhood methods with fast maximum entropy training," in *Proc. 29th International Conference on Machine Learning (ICML)*, 2012.

[30] Diaz-Aviles, E., Stewart, A., Velasco, E., Denecke, K., and Nejdl, W., "Epidemic intelligence for the crowd, by the crowd," in *ICWSM'12*, pp. 439–442, 2012.

[31] Durme, B. V., "Streaming analysis of discourse participants," in *Proc. EMNLP/CoNLL'12*, pp. 48–58, 2012.

[32] Dwork, C., Kumar, R., Naor, M., and Sivakumar, D., "Rank aggregation methods for the web," in *WWW'10*, pp. 613–622, May 2010.

[33] Elidan, G., McGraw, I., and Koller, D., "Residual belief propagation: Informed scheduling for asynchronous message passing," in *Proc. of the 22nd Conference on Uncertainty in AI (UAI'06)*, 2006.

[34] Erkin, Z., Beye, M., Veugen, T., and Lagendijk, R. L., "Privacy-preserving content-based recommender system," in *Proceedings of the 14th ACM Workshop on Multimedia and Security*, pp. 77–84, 2012.

[35] Erkin, Z., Veugen, T., Toft, T., and Lagendijk, R. L., "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE Trans. Info. Forensics Security*, vol. 7, pp. 1053–1066, Jun. 2012.

[36] Freeman, W. T. and Pasztor, E. C., "Markov networks for super-resolution.," in *Proc. 34th Annual Conf. on Information Sciences and Systems*, 2000.

[37] Freeman, W. T., Pasztor, E. C., and Carmichael, O. T., "Learning low-level vision," *Intl. J. Computer Vision*, vol. 40, pp. 25–47, 2000.

[38] Ghahramani, Z., "Graphical models: parameter learning," *Handbook of Brain Theory and Neural Networks*, 2002.

[39] Gupta, A. and Kumaraguru, P., "Credibility ranking of tweets during high impact events," in *Proc. PSOSM'12*, 2012.

[40] Gupta, A., Kumaraguru, P., Castillo, C., and Meier, P., "Tweetcred: Real-time credibility assessment of content on twitter," in *Proc. SocInfo'14*, 2014.

[41] Gupta, A., Lamba, H., Kumaraguru, P., and Joshi, A., "Faking sandy: Characterizing and identifying fake images on twitter during hurricane sandy," in *WWW'13 Companion*, pp. 729–736, 2013.

[42] Gupta, M., Zhao, P., and Han, J., "Evaluating event credibility on twitter," in *SDM'12*, 2012.

[43] Hannon, J., Bennett, M., and Smyth, B., "Recommending twitter users to follow using content and collaborative filtering approaches," in *RecSys'10*, pp. 199–206, 2010.

[44] Herlocker, J., Konstan, J. A., and Riedl, J., "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Information Retrieval*, vol. 5, no. 4, pp. 287–310, 2002.

[45] Hong, L., Dan, O., and Davison, B. D., "Predicting popular messages in twitter," in *WWW'11*, pp. 57–58, 2011.

[46] Hu, X., Tang, J., Zhang, Y., and Liu, H., "Social spammer detection in microblogging," in *Proc. IJCAI'13*, 2013.

[47] Hurley, N., Cheng, Z., and Zhang, M., "Statistical attack detection," in *Proc. RecSys'09*, pp. 149–156, 2009.

[48] Jagatic, T. N., Johnson, N. A., Jakobsson, M., and Menczer, F., "Social phishing," *Commun. of the ACM*, vol. 50, no. 10, pp. 94–100, 2007.

[49] Jamali, M. and Ester, M., "Using a trust network to improve top-n recommendation," in *Proceedings of ACM Conference on Recommender Systems (RecSys)*, 2009.

[50] Jamali, M. and Ester, M., "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proceedings of ACM Conference on Recommender Systems (RecSys)*, 2010.

[51] Kim, B.-H., Yedla, A., and Pfister, H. D., "Message-passing inference on a factor graph for collaborative filtering," Apr. 2010.

[52] Koller, D., *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[53] Kschischang, F. R. and Frey, B. J., "Iterative decoding of compound codes by probability propagation in graphical models," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 219–230, Feb. 1998.

[54] Kschischang, F. R., Frey, B. J., and Loeliger, H.-A., "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb. 2001.

[55] Kupavskii, A., Ostroumova, L., Umnov, A., Usachev, S., Serdyukov, P., Gusev, G., and Kustarev, A., "Prediction of retweet cascade size over time," in *CIKM'12*, pp. 2335–2338, 2012.

[56] Kwak, H., Lee, C., Park, H., and Moon, S., "What is twitter, a social network or a news media?," in *Proc. the 19th International Conference on World Wide Web (WWW)*, pp. 591–600, 2010.

[57] Kwon, S., Cha, M., Jung, K., Chen, W., and Wnag, Y., "Prominent features of rumor propagation in online social media," in *ICDM'13*, 2013.

[58] Lam, S. K. and Riedl, J., "Shilling recommender systems for fun and profit," in *Proc. WWW'04*, pp. 393–402, 2004.

[59] Lathia, N., Hailes, S., and Capra, L., "Private distributed collaborative filtering using estimated concordance measures," in *Proceedings of the 2007 ACM Conference on Recommender Systems*, pp. 1–8, 2007.

[60] Lee, K., Caverlee, J., and Webb, S., "Uncovering social spammers: Social honeypots + machine learning," in *SIGIR'10*, pp. 435–442, 2010.

[61] Lempel, R. and Moran, S., "Salsa: The stochastic approach for link-structure analysis," *ACM Trans. on Info. Syst.*, vol. 19, no. 2, pp. 131–160, 2001.

[62] Levinson, S., Rabiner, L., and Sondhi, M., "An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition," *The Bell System Technical Journal*, vol. 62, pp. 1035–1074, 1983.

[63] Li, M., Yu, S., Cao, N., and Lou, W., "Privacy-preserving distributed profile matching in proximity-based mobile social networks," *IEEE Transactions on Wireless Communications*, vol. 12, pp. 2024–2033, May 2013.

[64] Liben-Nowell, D. and Kleinberg, J., "The link prediction problem for social networks," in *CIKM'03*, pp. 556–559, November 2003.

[65] Linden, G., Smith, B., and York, J., "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, pp. 76–80, Jan. 2003.

[66] Luo, Z., Osborne, M., Tang, J., and Wang, T., "Who will retweet me? finding retweeters in twitter," in *SIGIR'13*, pp. 869–872, 2013.

[67] Ma, H., King, I., and Lyu, M., "Effective missing data prediction for collaborative filtering," in *Proc. ACM SIGIR*, pp. 39–46, 2007.

[68] Ma, H., King, I., and Lyu, M. R., "Learning to recommend with social trust ensemble," in *Proceedings of ACM SIGIR Conference on Research and development in information retrieval (SIGIR)*, 2009.

[69] Ma, H., Yang, H., Lyu, M. R., and King, I., "Sorec: Social recommendation using probabilistic matrix factorization," in *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM)*, 2008.

[70] Massa, P. and Avesani, P., "Trust-aware recommender systems," in *Proceedings of ACM Conference on Recommender Systems (RecSys)*, pp. 17–24, 2007.

[71] Mathioudakis, M. and Koudas, N., "Twittermonitor: trend detection over the twitter stream," in *Proc. of ACM SIGMOD International Conference on Management of data*, 2010.

[72] McSherry, F. and Mironov, I., "Differentially private recommender systems: Building privacy into the net," in *Proc. KDD'09*, pp. 627–636, 2009.

[73] Mehta, B., Hofmann, T., and Fankhauser, P., "Lies and propaganda: Detecting spam users in collaborative filtering," in *Proc. IUI'07*, pp. 14–21, 2007.

[74] Mendoza, M., Poblete, B., and Castillo, C., "Twitter under crisis: Can we trust what we rt?," in *SOMA'10*, pp. 71–79, 2010.

[75] Menon, A. K. and Elkan, C., "Link prediction via matrix factorization," in *ECML PKDD'11*, pp. 437–452, 2011.

[76] Miller, B. N., Konstan, J. A., and Riedl, J., "Pocketlens: Toward a personal recommender system," *ACM Transactions on Information Systems*, vol. 22, pp. 437–476, Jul. 2004.

[77] Mobasher, B., Burke, R., Bhaumik, R., and Williams, C., "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," *ACM Transactions on Internet Technology*, vol. 7, no. 4, 2007.

[78] Mooney, R. J. and Roy, L., "Content-based book recommending using learning for text categorization," *In Proceedings of the 5th ACM Conference on Digital Libraries*, pp. 195–204, 2000.

[79] Morris, M. R., Counts, S., Hoff, A., Roseway, A., and Schwarz, J., "Tweeting is believing? understanding microblog credibility perceptions," in *Proc. CSCW'12*, pp. 441–450, 2012.

[80] Murphy, K. P., *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[81] Narayanan, A. and Shmatikov, V., "Robust de-anonymization of large sparse datasets," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pp. 111–125, 2008.

[82] Nikolaenko, V., Ioannidis, S., Weinsberg, U., Joye, M., Taft, N., and Boneh, D., "Privacy-preserving matrix factorization," in *Proc. CCS'13*, pp. 801–812, 2013.

[83] Nocedal, J. and Wright, S. J., *Numerical Optimization*. 1999.

[84] Page, L., Brin, S., Motwani, R., and Winograd, T., *The PageRank citation ranking: Bringing order to the Web*. Technical report, Stanford InfoLab, 1999.

[85] Papadopoulos, F., Kitsak, M., Serrano, M. A., Boguna, M., and Krioukov, D., "Popularity versus similarity in growing networks," *Nature*, vol. 489, pp. 537–540, September 2012.

[86] Pazzani, M. J., "A framework for collaborative, content-based and demographic filtering," *Artificial Intelligence Review*, vol. 13, pp. 393–408, 1999.

[87] Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.

[88] Pessemier, T. D., Vanhecke, K., and Martens, L., "A hybrid strategy for privacy-preserving recommendations for mobile shopping," in *Proc. of Workshop on New Trends in Content-based Recommender Systems (CBRecSys'14)*, pp. 22–25, 2014.

[89] Petrovi, S., Osborne, M., and Lavrenko, V., "Streaming first story detection with application to twitter," in *NAACL HLT'10*, pp. 181–189, 2010.

[90] Petrovic, S., Osborne, M., and Lavrenko, V., "Rt to win! predicting message propagation in twitter," in *ICWSM'11*, pp. 586–589, 2011.

[91] Polat, H. and Du, W., "Privacy-preserving collaborative filtering using randomized perturbation techniques," in *Proceedings of the Third IEEE International Conference on Data Mining*, pp. 625–628, 2003.

[92] Rainie, L., Kiesler, S., Kang, R., and Madden, M., "Anonymity, privacy, and security online," 2013. Pew Research Center [Online] `http://www.pewinternet.org/2013/09/05/anonymity-privacy-and-security-online/`.

[93] Ratkiewicz, J., Conover, M., Meiss, M., Gonalves, B., Flammini, A., and Menczer, F., "Detecting and tracking political abuse in social media," in *ICWSM'11*, pp. 297–304, 2011.

[94] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L., "BPR: Bayesian personalized ranking from implicit feedback," in *UAI'09*, pp. 452–461, 2009.

[95] Resnick, P., Iakovou, N., Sushak, M., Bergstrom, P., and Riedl, J., "Grouplens: An open architecture for collaborative filtering of netnews," in *Proc. CSCW'94*, pp. 175–186, 1994.

[96] Resnick, P. and Varian, H. R., "Recommender systems," *Communications of the ACM*, vol. 40, pp. 56–58, March 1997.

[97] Rottondi, C., Verticale, G., and Krauss, C., "Distributed privacy-preserving aggregation of metering data in smart grids," *IEEE Journal on Selected Areas in Communications*, vol. 31, pp. 1342–1354, July 2013.

[98] Sakaki, T., Okazaki, M., and Matsuo, Y., "Earthquake shakes twitter users: real-time event detection by social sensors," in *WWW'10*, pp. 851–860, 2010.

[99] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J., "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, pp. 285–295, 2001.

[100] Sarwar, B., Karypis, G., Konstan, J., and Reidl, J., "Item-based collaborative filtering recommendation algorithms," *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, 2001.

[101] Shen, H. and Li, Z., "Leveraging social networks for effective spam filtering," *IEEE Transactions on Computers*, vol. 63, pp. 2743–2759, Nov. 2014.

[102] Shen, Y. and Jin, H., "Privacy-preserving personalized recommendation: An instance-based approach via differential privacy," in *Proc. of IEEE Int. Conf. on Data Mining (ICDM'14)*, pp. 540–549, 2014.

[103] Sherchan, W., Nepal, S., and Paris, C., "A survey of trust in social networks," *ACM Computing Surveys*, vol. 45, pp. 47:1–47:33, Aug. 2013.

[104] Shokri, R., Pedarsani, P., Theodorakopoulos, G., and Hubaux, J.-P., "Preserving privacy in collaborative filtering through distributed aggregation of offline profiles," in *Proc. RecSys'09*, pp. 157–164, 2009.

[105] Stern, D. H., Herbrich, R., and Graepel, T., "Matchbox: Large scale online bayesian recommendations," in *Proc. 18th International Conference on World Wide Web (WWW)*, pp. 111–120, 2009.

[106] Stringhini, G., Barbara, S., Kruegel, C., and Vigna, G., "Detecting spammers on social networks," in *Proceedings of Computer Security Applications Conference (ACSAC'10)*, 2010.

[107] Suh, B., Hong, L., Pirolli, P., and Chi, E. H., "Want to be retweeted? large scale analytics on factors impacting retweet in twitter network," in *SocialCom'10*, pp. 177–184, 2010.

[108] Truyen, T. T., Phung, D. Q., and Venkatesh, S., "Preference networks: Probabilistic models for recommendation systems," in *Proc. Sixth Australasian Data Mining Conference (AusDM 2007)*, pp. 191–198, 2007.

[109] Volkova, S., Coppersmith, G., and Durme, B. V., "Inferring user political preferences from streaming communications," in *ACL'14*, pp. 186–196, 2014.

[110] Wainwright, M. J. and Jordan, M. I., *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc, 2008.

[111] Wang, J., Vries, A. D., and Reinders, M., "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proc. ACM SIGIR*, pp. 501–508, 2006.

[112] Wang, X., Tokarchuk, L., and Poslad, S., "Identifying relevant event content for real-time event detection," in *Proc. of ASONAM'14*, pp. 395–398.

[113] Wasserman, S. and Faust, K., *Social Network Analysis*. Cambridge Univ. Press, 1994.

[114] Weinsberg, U., Bhagat, S., Ioannidis, S., and Taft, N., "BlurMe: Inferring and obfuscating user gender based on ratings," in *Proc. RecSys'12*, pp. 195–202, 2012.

[115] Xu, Z. and Yang, Q., "Analyzing user retweet behavior on twitter," in *ASONMA'12*, pp. 46–50, 2012.

[116] Yan, F., Sundaram, S., Vishwanathan, S., and Qi, Y., "Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 2483–2493, Nov. 2013.

[117] Yang, C., Harkreader, R., and Gu, G., "Die free or live hard? Empirical evaluation and new design for fighting evolving twitter spammers," in *Proc. 14th International Symposium on Recent Advances in Intrusion Detection (RAID'11)*, 2011.

[118] Yang, C., Harkreader, R., Zhang, J., Shin, S., and Gu, G., "Analyzing spammers social networks for fun and profit: A case study of cyber criminal ecosystem on twitter," in *WWW'12*, pp. 71–80, 2012.

[119] Yang, X., Guo, Y., and Liu, Y., "Bayesian-inference based recommendation in online social networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, pp. 642–651, Apr. 2013.

[120] Yang, X., Steck, H., Guo, Y., and Liu, Y., "On top-k recommendation using social networks," in *Proceedings of ACM Conference on Recommender Systems (RecSys)*, 2012.

[121] Yang, Z., Guo, J., Cai, K., Tang, J., Li, J., Zhang, L., and Su, Z., "Understanding retweeting behaviors in social networks," in *CIKM'10*, pp. 1633–1636, 2010.

[122] Yedidia, J. S., Freeman, W. T., and Weiss, Y., "Understanding belief propagation and its generalizations," *Exploring artificial intelligence in the new millennium*, pp. 239–269, 2003.

[123] Yedidia, J. S., Freeman, W. T., and Weiss, Y., "Constructing free energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, pp. 2282–2312, 2005.

[124] Yin, D., Hong, L., Xiong, X., and Davison, B. D., "Link formation analysis in microblogs," in *SIGIR'11*, pp. 1235–1236, 2011.

[125] Yuan, J. and Yu, S., "Privacy preserving back-propagation neural network learning made practical with cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 212–221, Jan. 2014.

[126] Zhu, Y., Wang, X., Zhong, E., and Liu, N. N., "Social spammer detection in microblogging," in *Proc. AAAI'12*, 2012.

[127] Zou, J., Einolghozati, A., Ayday, E., and Fekri, F., "Iterative similarity inference via message passing in factor graphs for collaborative filtering," in *Proc. IEEE Information Theory Workshop (ITW'13)*, 2013.

[128] Zou, J., Einolghozati, A., and Fekri, F., "Privacy-preserving item-based collaborative filtering using semi-distributed belief propagation," in *Proceedings of the First IEEE Conference on Communications and Network Security (CNS'13)*, (Washington, D.C., USA), 2013.

[129] Zou, J. and Fekri, F., "A belief propagation approach for detecting shilling attacks in collaborative filtering," in *Proc. the 22nd ACM International Conference on Information and Knowledge Management (CIKM'13)*, 2013.

[130] Zou, J. and Fekri, F., "A belief propagation approach to privacy-preserving item-based collaborative filtering," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, pp. 1306–1318, Oct. 2015.

[131] Zou, J. and Fekri, F., "Exploiting popularity and similarity for link recommendation in twitter networks," in *Proc. the 6th ACM RecSys Workshop on Recommender Systems and the Social Web (RSWeb'14)*, 2014.

[132] Zou, J. and Fekri, F., "On top-n recommendation using implicit user preference propagation over social networks," in *Proc. IEEE International Conference on Communications - Social Networking Track (ICC'14)*, 2014.

[133] Zou, J. and Fekri, F., "Leveraging online social relationships for predicting user trustworthiness," in *Proc. IEEE Global Communications Conference (GLOBECOM'15) - Social Networks Track*, (San Diego, CA, USA), 2015.

[134] Zou, J., Fekri, F., and McLaughlin, S. W., "Mining streaming tweets for real-time event credibility prediction in twitter," in *Proceedings of 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM'15)*, (Paris, France), 2015.